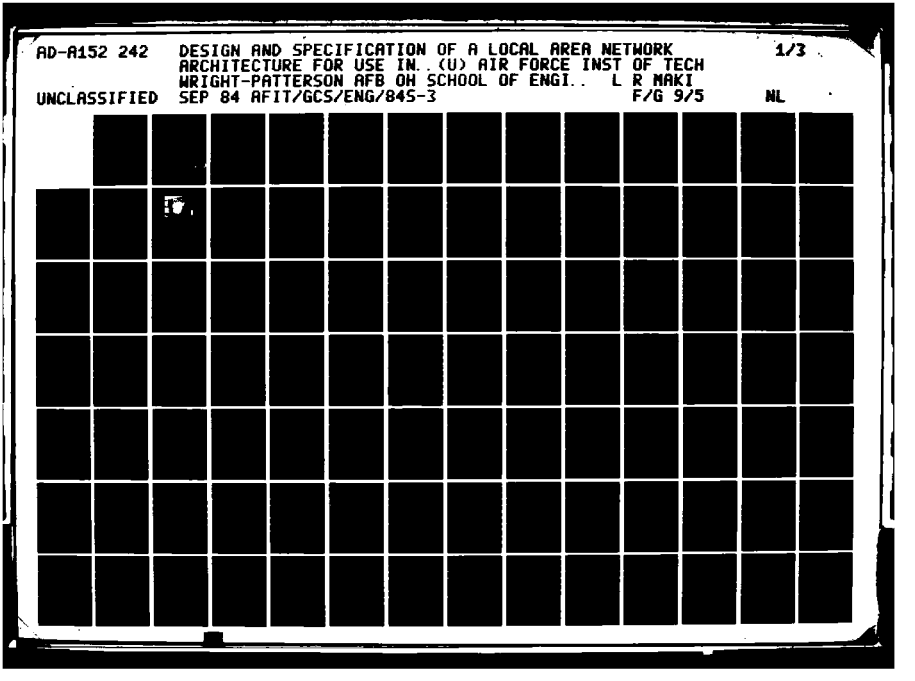
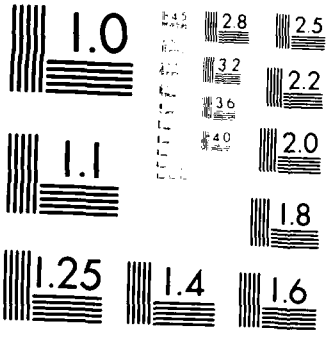
AD-A152 242 DESIGN AND SPECIFICATION OF A LOCAL AREA NETWORK ARCHITECTURE FOR USE IN.. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. L R MAKI

1/3

UNCLASSIFIED SEP 84 AFIT/GCS/ENG/84S-3 F/G 9/5 NL



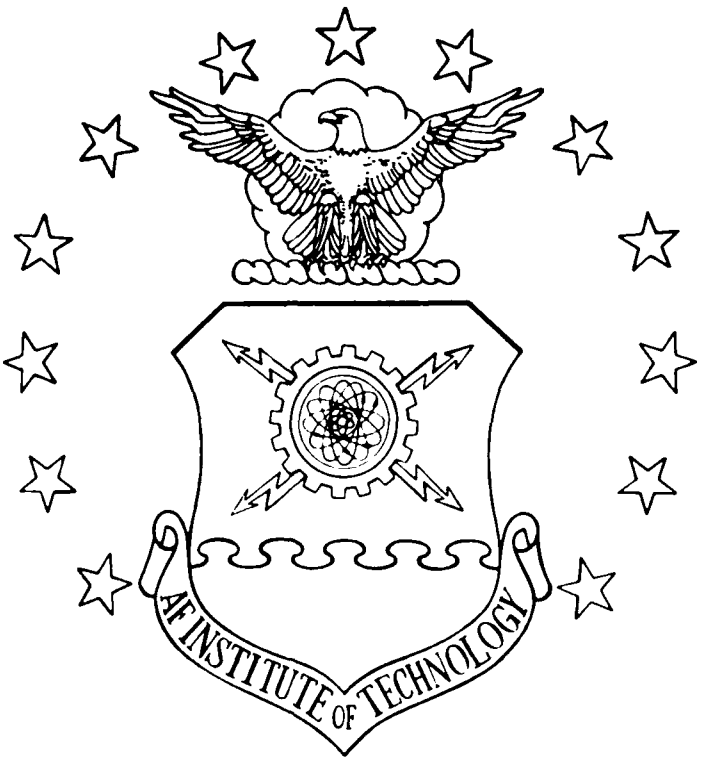


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

REPRODUCED AT GOVERNMENT EXPENSE

DTIC

AD-A152 242



DESIGN AND SPECIFICATION OF A LOCAL AREA NETWORK ARCHITECTURE FOR USE IN REAL-TIME FLIGHT SIMULATION

THESIS

Luke R. Maki
GS-12

AFIT/GCS/ENG/84S-3

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

DTIC
ELECTE
APR 9 1985
S D
B

DTIC FILE COPY

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

85 08 13 112

AFIT/GCS/ENG/84S-3

DESIGN AND SPECIFICATION OF A LOCAL AREA NETWORK ARCHITECTURE FOR USE IN REAL-TIME FLIGHT SIMULATION

THESIS

Luke R. Maki
GS-12

AFIT/GCS/ENG/84S-3

Approved for public release; distribution unlimited

DTIC
ELECTE
S APR 9 1985 D
B

AFIT/GCS/ENG/84S-3

DESIGN AND SPECIFICATION OF A LOCAL AREA NETWORK ARCHITECTURE FOR USE IN REAL-TIME FLIGHT SIMULATION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Systems

Luke R. Maki, B.S.

GS-12

September 1984

Approved for public release; distribution unlimited

Acknowledgements

In performing the research for and writing of this thesis I have had a tremendous amount of help from others. I wish to thank my faculty advisor, Maj Walter D. Seward, for his guidance and assistance during the entire course of my thesis effort. I am also indebted to my thesis committee member Capt John Gordon for the time he spent reviewing draft after draft. For technical assistance, thanks go to Capt A. Jay Kirchoff and Lt Richard P. Benken. Thanks are also due Mr Paul Blatt and Mr Bill Klotzback of the Flight Dynamics Lab for their support and effort in sponsoring and funding this effort, and for their useful comments in the final written stages of the thesis. For producing the final typed copy of this thesis in record time, my thanks go to Linda Clere. A word of thanks is owed to the Kirchoffs (Jay and Sue) for allowing me to take over the office and guest room in their home during the final two months of my thesis effort. Finally, I wish to thank my wife Faren for her understanding and patience during those same two months as she awaited my arrival to our new home in Seattle.

Luke R. Maki

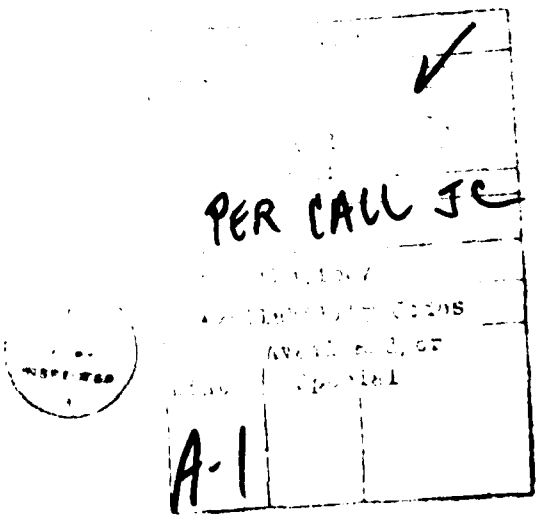


Table of Contents

List of Figures

List of Tables

Abstract

This investigation examined the use of a Local Area Network (LAN) in the real-time environment of the Air Force Flight Dynamics Laboratory's Flight Control Development Laboratory (FCDL) flight simulation facility.

Using the requirements of the FCDL's Manned Combat Station (MCS) project as a guideline, a LAN based on the Ethernet protocol specification was identified as suitable for use in the real-time flight simulation facility. Both the architecture and the performance of the Ethernet protocol were examined to make this conclusion. The selection of the Intel 82586 Local Communications Controller chip (which defaults to the Ethernet specification), the in-house design and manufacture of a Network Interface Board (NIB) using this chip, and the standard operating procedure of FCDL simulations allowed a software design to be completed for the control of the NIB. The investigation is concluded with a recommended course of action for the eventual implementation of the proposed LAN and software design.

DESIGN AND SPECIFICATION OF A LOCAL AREA NETWORK ARCHITECTURE FOR USE IN REAL-TIME FLIGHT SIMULATION

I. Introduction

The Flight Control Development Laboratory (FCDL) is a research and development aircraft simulation facility supporting the Flight Control Division of the USAF Flight Dynamics Laboratory. Within the FCDL, a need exists to continually update and improve the simulation facility. This includes the enhancement of both the simulation computational equipment as well as the aircraft simulator cockpits and associated hardware. In the past, when a new piece of equipment was installed in the facility, a highly specialized, one-of-a-kind interface was developed to link new equipment with existing equipment. Each new peripheral would use an interface that was customized for that particular application. Often, these interfaces would be developed by different vendors with different design philosophies. The result has been a collection of specialized interfaces that have proved to be very expensive, difficult to maintain, and often fail to meet desired transfer rates.

The FCDL facility is planning to install several new peripheral processors in the facility. This installation is being accomplished by in-house personnel. With this installation will come a new interfacing concept that is proposed to be the FCDL standard for linking peripherals

with the simulation computers. The proposed concept is the implementation of a Local Area Network (LAN) of simulation peripherals. The distinguishing feature of the proposed LAN is its use for real-time aircraft simulation.

The phrase "real-time" needs additional explanation. Flight simulations conducted in the FCDL are primarily research and development programs to investigate advanced flight technologies, with a man-in-the-loop to allow the man-machine interactions to be analyzed. These simulations are achieved by mathematically modeling an aircraft on a hybrid system for digital and analog computers, and are considered "real-time" in that the simulated aircraft states match those of the real aircraft at any point in time for the same environmental and pilot input conditions. Thus, the pilot experiences the illusion of actually flying the aircraft being simulated. Since digital computers are used, the states of the aircraft are computed once every cycle. This cycle time is generally on the order of 25 milliseconds for simulations in the FCDL. For a LAN to be considered "real-time", then, two or more computers must be able to exchange information at least once every cycle time. As will be shown in Chapter IV, real-time operation can be obtained with an appropriate LAN.

Several factors have generated the need for the implementation of this type of LAN. First, the number of expansion slots available in the main simulation computers, Systems Engineering Laboratory (SEL) 32/7780's, is limited. Only one more interface board can be added to each of the two computers and, without a LAN, this means each could communicate with only one peripheral per expansion slot. With the

implementation of a LAN, the main simulation computers can communicate with many peripherals while using only one of these expansion slots.

Second, it is becoming apparent that the FCDL simulation facility will be required to interface its simulation computers to existing peripherals in other branches of the Flight Control Division. Those peripherals will have their own intelligent controllers and will be used on a temporary basis for a particular simulation, then returned to the appropriate branch. The use of a LAN allows for this temporary expansion without developing entirely new interfaces to the SEL simulation computers. The LAN provides a standard protocol and technique for communicating between the simulation computers and peripheral controllers.

Third, the FCDL has an immediate need to interface the main simulation computer to a new peripheral processor for the Manned Combat Station (MCS) program. Part of the MCS program involves interfacing a microprocessor controlled workstation to the main simulation computers. Another portion of the program involves researching the use of distributed microprocessor controllers for I/O control of simulator cockpits. For example, an individual cockpit controller would control cockpit instrumentation and compute the aerodynamics model for a simulated aircraft, while possibly interfacing real-time with one or more other simulations attached to the network. To coordinate these simulations and to make the pilot of one simulation aware of another's existence, the state information of each must be transferred over the network to the other. Other logical information, such as the state of the

megabit/second to make the transmission rates of the network components somewhat compatible, and yet keep a low cost per node option viable.

The fourth protocol requirement above is necessary to make sure the flexibility of the FCDL research facility remains intact. Data of variable length and of variable type (real, integer, logical) can be expected to be transferred. Furthermore, the order of data in a data packet may change from simulation to simulation.

Finally, the fifth protocol requirement above is necessary in terms of simulation fidelity, accuracy, and safety. An error in the data transferred over a network in a real-time, man-in-the-loop flight simulation could cause a discontinuity and, if the discontinuity is in a motion command to a moving-base simulator, physical harm could result to the pilot.

Furthermore, minimal operational requirements for use on the SEL include:

1) Any data handler software developed in support of the protocol design will be implemented in SEL Assembly language;

2) The protocol software will be accessible to the user via a FORTRAN library function.

The first operational requirement above is necessary as there is no Higher-Order-Level language available with which to either develop the data handler software or run the software. Therefore, the data handler must be written in SEL Assembly language.

The second operational requirement above will allow the user to simply call upon the data handler as a function. Standardization and ease of use were the primary motives for establishing this requirement.

is available. In other words, transmissions can occur at unknown times (at least during the first cycle) during a simulation cycle time because the different stations will be accessing the network at random times, dependent upon when each station requests network time.

The third protocol requirement above was specified after surveying the data transfer rates of commercially available LAN's, and determining that the HSD might be the slowest element in the data communications chain. The HSD is advertised to have a maximum data rate of 834 kilowords/second, where a word is four bytes long to match the bus of the SEL. In keeping with the intent of being able to interface any processor to the network, with 16 bits being the most likely, a nominal 16 bits was selected as the design point. Therefore, the HSD would have a maximum data rate of

$$(834 \text{ kilowords/second})\,(2 \text{ bytes/word})\,(8 \text{ bits/byte}) = 13.344 \text{ megabits/second (Mb/s)} \quad (1)$$

with overhead not included. Including an assumed overhead factor of 1.5, this rate drops to 8.896 Mb/s. A 10 Mb/s Ethernet system using 500 byte data packets (the size anticipated for use by the FCDL, discussed in Chapter IV) has overhead reducing the transfer rate of usable data to 9.294 Mb/s (18 byte header, 8 byte preamble, and an interframe spacing of 9.6 microseconds). The survey (see Table I, Chapter III) indicated that 10 megabit/second systems were the most common, with 12 and 50 megabit/second systems available. The latter two, however, were not attractive due to their high cost relative to the cost of some of the 10 megabit/second systems. The requirement was thus placed at 10

simulations. The user's requirements for the network are subtly stated in the above excerpt. More succinct statements of the requirements follow, with some background explanation as to their origin.

Baseline Effort

The minimal requirements for the real-time LAN to be acceptable for FCDL simulation research can be defined in two areas, namely protocol and hardware. The minimization of LAN cost is also of concern as the FCDL operates within a limited budget.

The protocol must have the following characteristics:

1) The data transfer must be conducted real-time to maintain the integrity of the man-in-the-loop simulation;

2) Data transmission must be conducted asynchronously;

3) The data transfer rate (the bandwidth of the medium) must be at least 10 megabits/second;

4) The protocol must be able to handle data in a "freefield" format; that is, data of variable length and of variable type;

5) The protocol must have error detection capability and appropriate methods to deal with an error.

The first protocol requirement above is obviously necessary for the facility to continue real-time flight simulations. This requirement places constraints on the amount of delay a data transmission can incur over the network. Thus, the network chosen and designed must be able to transfer all required data within the cycle time of the simulation.

The second protocol requirement above refers to the contention that the SEL, or any other processor, will not communicate directly with the network via handshaking mechanisms. Rather, the processor will request that data be transmitted, and the NIB will transmit it when the network

allows them to provide more computational power. This use of a microprocessor to control the cockpit has been coined a "smart cockpit". In order to implement the concept of a "smart cockpit", a technique must be developed to communicate real-time between the simulation computer and this microprocessor controller. This communication link must be able to also communicate to other intelligent devices that may be added to this communications link at a future date. . .

. . .The network interface must be a real-time, high-speed, bidirectional, digital-to-digital link between the central simulation computer (a SEL 32/77) and a microprocessor-based MCS cockpit controller. The interface will allow the transfer of data between the two systems in order to relieve the SEL of some of its computational load, and to supply the SEL with, for example, the state of the MCS simulation for relative aircraft computations. An expansion of the interface to allow more than one peripheral processor will be addressed later in the project.

The network interface effort will be conducted in two tasks, the second utilizing the results of the first. The primary task is to develop the protocol and hardware on the SEL necessary to communicate with one peripheral microprocessor real-time. The protocol must have effort detection capability to assure integrity of the simulation, with appropriate contingencies in the event of an error. The minimum data transfer rate must be 10 megabits/second. The hardware must be designed such that communications with the peripheral processor are possible over a maximum distance of 350 feet.

The second task will address the issue of real-time data communication with more than one peripheral processor. This task will involve more testing than designing as the protocol necessary will be anticipated and included in the first task. The data transfer rate and error detection requirements must be maintained. the results of these tests will have a tremendous impact on the design of future networked simulations.

The overall intent of the MCS project, then, is to first establish a functioning LAN with two nodes in order to prove the concept of using a LAN for real-time communications. The ultimate objective is to expand the network to include additional nodes for more complex M-on-N

II. User Requirements for Network

The requirements of the real-time LAN for any user in the FCDL are essentially the same as for the Manned Combat Station (MCS) project, and are generally stated in the MCS's requirements specification (Kirchoff and others, 1983). As such, the network interface specification described in this paper is an integral part of the MCS project, and must be defined to assure proper data communications between the hardware and the software of the integrated system.

The following is excerpted from the MCS requirements specification to provide further understanding:

> AFWAL/FIGD requires the immediate ability to simulate 1-on-1 piloted aircraft engagements, and will eventually require M-on-N piloted aircraft engagements capability, where M+N = 4. AFWAL/FIGD currently has the capability to investigate in detail most of the operations of total mission analysis. These investigations currently use an adaptive computer target model for the experiment pilot to "chase". The MCS is designed to replace this adaptive computer model with a piloted target aircraft. This MCS pilot will also be allowed to fully participate in the simulation and not act as just a chase vehicle. However, the MCS is not required to provide the simulation fidelity that currently exists in the cockpits within the Flight Control Development Laboratory. This reduction in fidelity will allow the introduction of a "simulation workstation" which allows a pilot to "fly" a fighter type aircraft without all cockpit instruments ad visual displays. Such a workstation has already been developed by McDonnell Douglas at their St. Louis simulation facility. This existing workstation, called a Manned Interactive Combat Station (MICS), will be used as a reference for the development of the AFWAL/FIGD Manned Combat Station. . .
>
> . . .The MCS will be designed to include a microprocessor-based computer system to increase flexibility to the workstation. The addition of computer intelligence to the MCS cockpit provides the ability to communicate from the simulation computers to the cockpit at a high instruction level rather than a raw data level. This offloads I/O from the simulation computers and

the communications protocol and hardware to be considered for the FCDL network. Chapter IV, through analytical studies, verifies that the protocol and hardware chosen can meet real-time operating requirements. Chapter V contains a brief description of the Intel 82586 network controller chip, and presents an interface software design for the FCDL network based on this chip and the Network Interface Board (NIB) designed in-house by Lt. Richard Benken of the organization AFWAL/FIGD (the Control Synthesis Branch of the Flight Control Division, USAF Flight Dynamics Laboratory). Control of the SEL 32/7780 HSD is also briefly considered. Finally, Chapter VI presents a summary of this work and contains a recommended course of action to be taken in following efforts.

the order of the transmitted data and assuring the destination is aware of that order. The Application layer will obviously be necessary, and will include tasks such as diagnostics.

Problem

The problem investigated in this study was to determine the hardware necessary to implement a real-time network in the FCDL based on FCDL requirements, analyze the expected performance of the network hardware, and design the protocol software necessary to allow an intelligent controller to obtain real-time communications on the network.

Scope

The hardware necessary to implement a real-time network for the FCDL was determined. An analysis was conducted to estimate the delays that would be incurred by using the proposed network, and the impact of such delays on simulation operation. The design of the protocol software is based on the Intel 82586 Local Communications Controller chip used on the NIB, and a suggested sequence of steps to be taken to initialize and control the chip is included. Also, a brief review of the High Speed Device Interface used with SEL 32/7780 is included for consideration by the user of the SEL on the proposed network.

Sequence of Presentation

After an introduction to the goals of the FCDL and the need for a Local Area Network are presented in Chapter I, Chapter II describes the requirements defined by the FCDL that the network must meet. Chapter III presents the results of a cost/capabilities study which determined

Finally, the Application layer directly serves the communicating end-user application process by providing the distributed information service appropriate to the application and its management.

Of the seven layers, the last four need to be considered for inclusion in software for the real-time application in the FCDL. The first two layers (Physical and Data Link) of the seven are being implemented in hardware and firmware for at least one communications protocol by several manufacturers (see Chapter III). Therefore, these layers can be obtained off-the-shelf. The third (Network) and fourth (Transport) layers are generally associated in their function of handling network control, such as addressing and routing information through the network, controlling errors, and accounting for services received. The most important function provide by these layers is the confirmation of packet receipt, a critically important function for a real-time application such as flight simulation. There must be assurances that a node on the network is updated every cycle, or the validity of the simulation may be in question. Note that, if a cyclic redundancy check (CRC) algorithm is implemented in the data link layer, the two layers complement one another by assuring that data is sent and that it is intact. The Session layer, as it deals with virtual circuits and the like, is not appropriate for the task at hand. The paths of communication between any two devices on the network will be simply determined by a destination address in the data packet. No special set-up procedures are anticipated. The Presentation layer will have to be considered only in that different types of data are anticipated to be transferred (real, integer, logical). The user will have the responsibility of managing

of the packet is passed up to the next layer for further examination as required.

A better understanding of the OSI model can be obtained with a brief description of the function of each layer. The Physical layer describes the physical media over which a bit stream is to be transmitted, and such specifics as cable type, signal levels, bit rate, etcetera.

The Data Link layer describes the rules for transmitting on the physical media, such as information format, the media access method, frame (information packet) transmission, and the media release mechanism. The latter two layers have been implemented on silicon chips for some popular protocols by several manufacturers, making the networker's job much easier (Burskyin, 1982: 80).

The Network layer governs the switching and routing of information between networks.

The Transport layer handles services such as end-to-end acknowledgements of successful message receptions, and other message integrity functions (besides functions such as cyclic redundancy code, or CRC, checks).

The Session layer manages the requesting and deleting of virtual circuit connection services provided by the transport layer.

The Presentation layer provides for any necessary translation, format conversion, or code conversion to put the information into a recognizable and useful form.

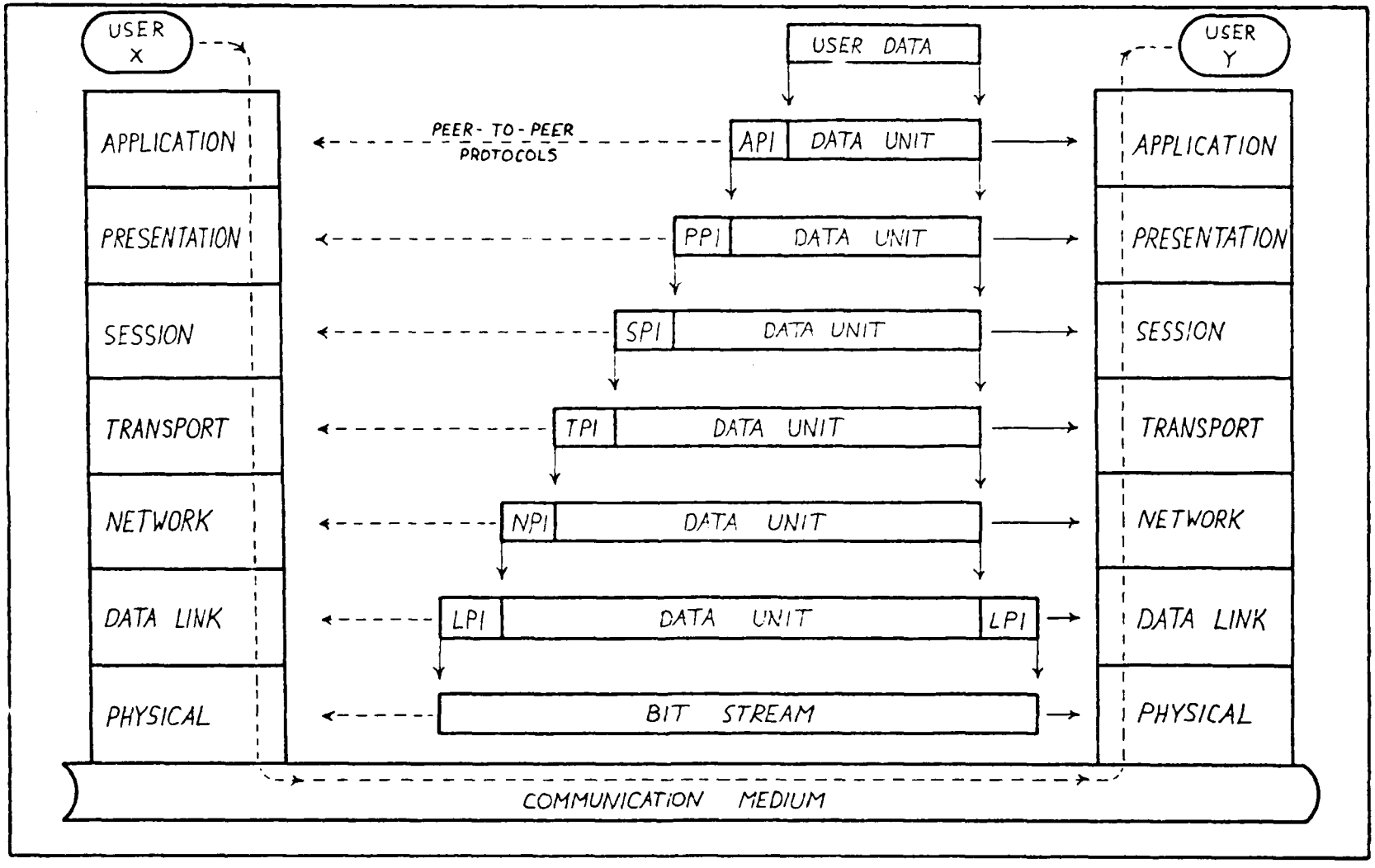


Figure 3. ISO Seven-Layer OSI Protocol Model

essentially a direct memory access (DMA) device which attaches to the SEL bus and controls I/O for the SEL 32/7780, thereby freeing the SEL CPU for other tasks.

The final item required is the network protocol to perform communications quickly and efficiently. The software designed to implement the protocol should be generic enough to allow easy portability between host computers, yet specific in its communication with the network interface.

As a way of directing the rapid growth of LAN's and their associated protocols, the International Standards Organization (ISO) drafted and the American National Standards Institute (ANSI) adopted an Open Systems Interconnection (OSI) reference model that now serves as a backbone for protocols within a network (Allan, 1982a: 107). As shown in Figure 3, one layer interacts with another via a protocol in three ways: peer to peer, where a layer communicates with the same layer of another node; service provider, where a lower layer provides a service to the next higher layer of the same node; and service requester, where a higher layer uses the services provided by the next lower layer of the same node. Information is passed from peer-to-peer by attaching a header to the beginning of the data packet that is to be transmitted (such as API, or Application protocol Interface header, as shown in Figure 3). Each layer tacks on its header as the packet descends through the layers. Each layer performs a service for the data packet based on control bits set in the packet. Upon receipt of the packet at the destination node, a similar process is performed in reverse order. Each layer examines the packet area where it expects to find its header. Any remaining part

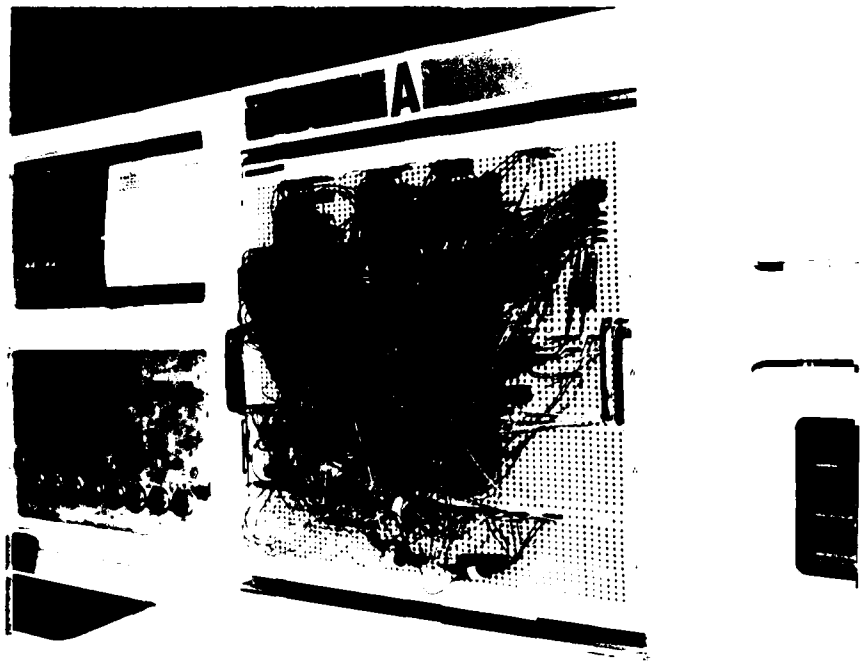


Figure 2. Typical Trunk Rack A Patching

communications with the main simulation computers. The same is true for other microcomputers procured to control additional peripheral equipment, such as the terrain boards.

Besides an intelligent controller, some interfacing hardware must exist to physically connect the peripheral to the network. One of the MCS program subtasks is to design, build, and test this network interface board, or NIB. The proposed design will operate with any parallel I/O interface under computer control. This generic approach allows one NIB design to operate for all nodes of the network. The FCDL SEL 32/7780 computers to be connected to the LAN have the additional requirement of using a SEL/Gould High Speed Device (HSD) intelligent input/output (I/O) peripheral to interface with the NIB. The HSD is

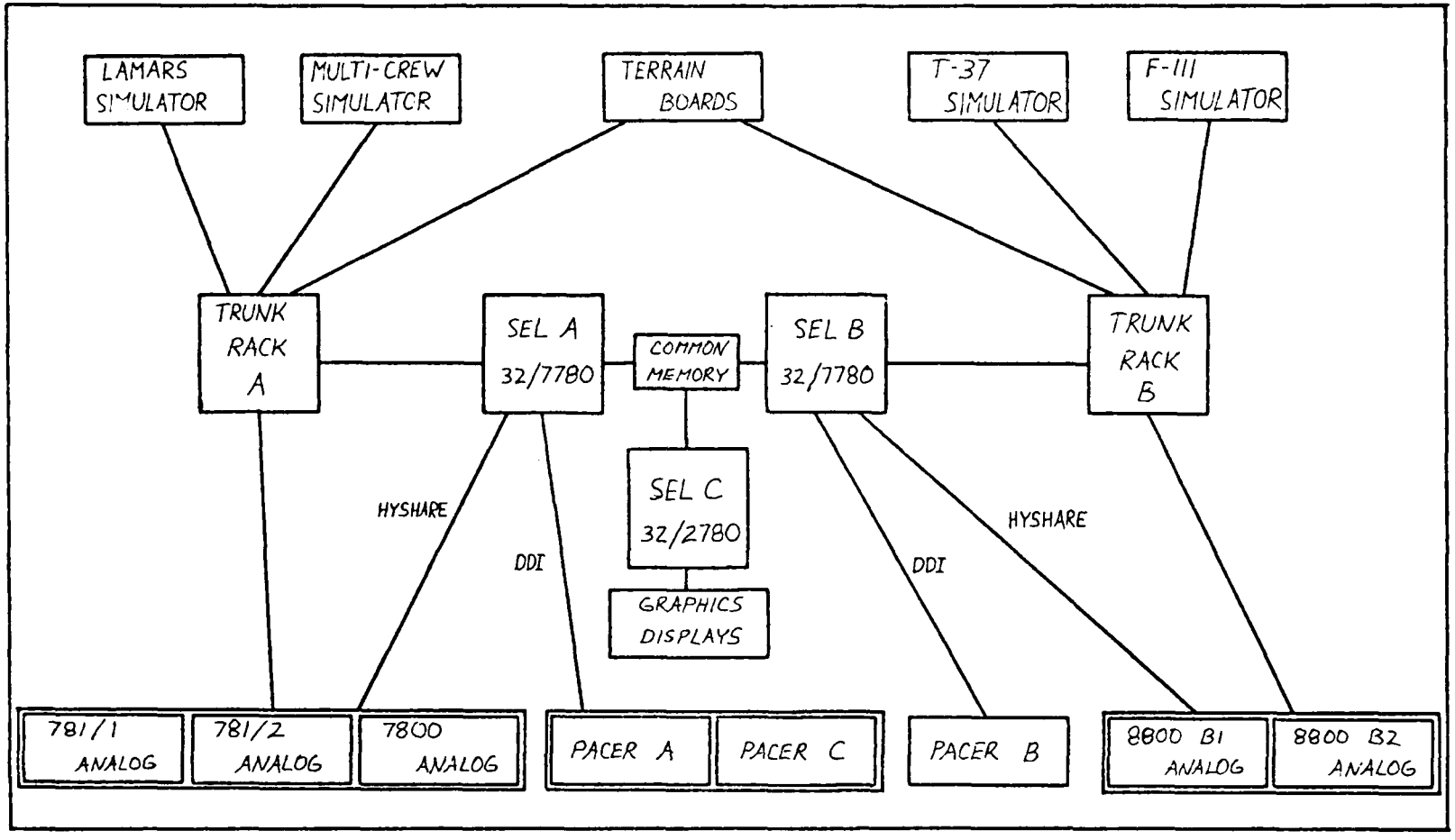


Figure 1. Present FCDL Computer System Configuration (Simplified)

simulation itself (Initial Conditions, IC; Operate, OP; Hold, HOLD; or Reset, RESET), must also be transferred. Should this use of distributed controllers prove to enhance simulation capabilities, the FCDL could expand the proposed LAN up to a dozen device controllers that would all be linked to the simulation computers. Because of this potential, the MCS program has chosen to develop a real-time communications LAN to link the MCS cockpit to the simulation computers.

A significant advantage with using a LAN is the elimination of the trunking stations (Trunk Racks A and B) that currently exist for the analog interfacing of simulation computers to the various simulation peripherals, including cockpits and terrain boards (Figure 1). A trunking station connects the appropriate computer outputs to the desired simulator inputs using patch panels and patch cords. Figure 2 is a picture of Trunk Rack A, indicating the complexity of the patching currently required. With the full expansion of the proposed real-time LAN, the cumbersome task of patching these panels, and the man-hours wasted trying to find broken wires and bent pins in a patched panel, can be avoided. Furthermore, the use of a LAN instead of trunk racks will minimize the distance analog signals must travel, and thereby minimize the noise they pick up along the way.

While the networking approach does provide good expansion potential beyond current interfacing techniques, there are equipment requirements that must be met before networking can satisfy FCDL's needs. To be connected to the network, a peripheral must have a user programmable, intelligent controller that communicates through a parallel I/O interface. For the case of the MCS, a microcomputer system will control all

The hardware must have the following characteristics:

1) Any new hardware will not require modifications to the existing SEL hardware;

2) The hardware must be able to interface with the parallel 32-bit data format of the SEL;

3) The hardware must be able to transmit the digital data over a 350 foot range with no errors caused by signal distortion.

The first hardware requirement above was made to force the network design to be generic and compatible with almost any system. The requirement as stated is really incomplete, and should read that no modifications are required to any system attached to the network. The intent is to be able to attach to the network any system that has a parallel port and some control lines.

The second hardware requirement above was made to assure the network could transmit all 32 bits of a four byte SEL word if necessary, even if the NIB interface was only 16 bits wide. The latter situation would just require two transfers to get the whole word transmitted (with the destination prepared to reassemble the two halves).

The third hardware requirement above was made based on an estimate that the furthest separation of any two nodes on the network would be 350 feet, or approximately 117 yards. A more accurate measurement revealed that 240 meters (787.4 feet, or 262.5 yards) is actually the minimum required (see Chapter III, Figure 4). Therefore, the design requirement stated in the MCS specification was too conservative. This change, however, does not impact the final result of this analysis.

Expanded Operation

Once the initial system LAN interface has been demonstrated, the goal will be to connect additional peripheral processors. The upper limit on the number of processors that could be connected to the network is practically bound by the data transfer rate, the amount of data transferred, and the amount of delay that can be tolerated in receiving the data. As will be shown in Chapter IV, the anticipated full complement of processors that will be connected to the network, twelve, will not significantly degrade response performance.

Summary

Using the MCS project as a guide, the user requirements for a real-time LAN have been stated in terms of network protocol requirements, hardware requirements, operational requirements, and cost constraints.

A summary of the requirements stated in this chapter appear below.

The protocol must have the following characteristics:

1) The data transfer must be conducted real-time;

2) Data transmission must be conducted asynchronously;

3) The data transfer rate must be at least 10 Mb/s (overhead not included);

4) The protocol must be able to handle "freefield" formatted data;

5) The protocol must have error handling capability.

Operational requirements include:

1) The data handler sotware for the SEL will be in Assembly language;

2) The protocol software will be accessible via a FORTRAN library function.

Hardware requirements include:

1) New hardware will not require modifications to existing SEL hardware;

2) The hardware must be able to interface with the parallel 32-bit data format of the SEL;

3) The hardware must be able to transmit digital data over a 240 meter range with no errors from signal distortion.

The selection of suitable network hardware to meet these requirements can now be made, and this topic is addressed in Chapter III.

III. Selection of Network Hardware

This chapter presents the approach used to select the hardware that would meet the requirements stated in Chapter II. Basically, a survey was performed of available networks that advertised a maximum data transfer rate (bandwidth of the media) of at least ten (10) megabits per second (Mb/s). Cost was next employed to narrow the choices, which resulted in one particular LAN protocol, known as Ethernet, being least expensive to implement per node. The suitability of this protocol or use in a real-time environment was examined first before further pursuing its use. This chapter discusses its suitability on physical operating characteristics, and Chapter IV discusses its operational performance. After determining it could be used based on physical operating characteristics, the hardware necessary for its implementation was selected.

Networks Meeting Data Transfer Rate Requirement

Table I below is a compilation of data (Allan, 1982: 92-93; Reagan, 1983: 4-121) which summarizes the local area networks available as of late 1982 meeting the data rate requirement. The networks of Table I represent a minority of the available networks providing data communication. Most available networks operate at a data rate less than ten Mb/s.

Least Costly Network

From Table I, it is obvious that there are many commercial networks available that meet the single requirement of ten Mb/s. However, the cost for attaching a user to a network varies by greater

TABLE I

Local Area Networks ≥ 10 Mb/s

Network Name	Manufacturer	Maximum Data Rate	Cost/Node
DOMAIN	Apollo Computer	12 Mb/s	$75,000 +
Ethernet	Xerox	10 "	$250 - $1,000
HYPERbus	Network Systems	10 "	$2,150 - $6,950
HYPERchannel	Network Systems	50 "	$40,000
Net/One	Ungerman-Bass	10 "	$500 - $2,500
Planet	Racal-Milgo	10 "	$1,000

than two orders of magnitude. From available data, the Ethernet network made by Xerox can be the least expensive at 250 dollars per connection, while connecting to Apollo Computer's DOMAIN network can cost in excess of 75,000 dollars. As it was FCDL management's intent to obtain this network at minimum cost, an Ethernet-type of network solution was decided upon as worthy of further investigation.

Suitability of Ethernet

Ethernet is a local area network developed jointly by Digital Equipment Corporation (DEC), Intel Corporation, and Xerox Corporation. The Ethernet specification (Digital Equipment Corporation and others, 1980) was developed through collaboration of the three corporations, and after several years of effort by Xerox on an earlier prototype Ethernet. The suitability of Ethernet for a real-time communications application such as for the MCS project can initially be determined by examining the design parameters characterizing the physical connection of elements

(microprocessors, terminals, etc.), or nodes, that make up the LAN. The primary parameters are: transmission mode, topology, transmission media, the class of elements to be interconnected, reliability, and, for implementation as soon as possible, availability. Performance characteristics, such as the delay a user might experience in the transfer of a data packet using the Ethernet protocol, are considered in Chapter IV.

Transmission mode refers to the use of either digital or analog signals on the network. The Ethernet is designed to transmit digital signals, and this is suitable for the intent of the FCDL LAN. It is expected that there will be analog-to-digital (A/D) and digital-to-analog (D/A) conversions necessary to conduct a flight simulation, but these operations will be done outside of the LAN network environment. For example, analog computers are presently used to model the flight control systems of simulated aircraft, but these are interfaced to the SEL 32/77's via the Hyshare link as shown in Figure 1. This arrangement will not change with the addition of a LAN. Also, hardware feedback signals from simulation equipment are typically analog, but these will be converted to digital by A/D converters at the equipment's host computer prior to being transmitted over the network. Therefore, an all-digital LAN meets requirements.

Topology of the network required in the FCDL is very important, as the transfer of data for real-time flight simulations will be the primary use. It is important that control of the simulation be distributed, in that the failure of any one element should not adversely effect the operation of other elements on the network. This constraint eliminates a star network as there is typically a primary element that

controls the operation of the whole network. Ring topologies are also not desirable as the failure of one element would preclude the continuation of a data message around the ring. It is important that data flow continue after a failure so that the simulation can be terminated in a controlled manner. A double ring network would allow continued data flow, but is accompanied by more complex hardware and protocol, and thus, higher cost. This leaves the bus topology, which is the most desirable as it is resistant to single-point failures (Franta and Chlamtac, 1981: 13). Ethernet has a bus topology, namely a branching non-rooted tree. This topology, as in other busses, allows the transmission of messages away from the originating node in both directions, and each other node can test message content as it passes by to determine if that node is the intended destination of a transmission. A failure of a node on such a system would still allow the transmission of messages between other nodes. The other nodes would become aware of the failed node during a simulation by either a controlled notification from the failed node, or the lack of transmissions from the failed node. In either case, the simulation could be terminated in a controlled manner.

Ethernet uses coaxial cable as the transmission medium, as most local networks do. Of the four most used media for any type of network (coaxial cable, radio, satellite, and fiber-optics), coaxial cable is the most popular due to its moderate cost, high bandwidth, and low bit error rate, typically between one bit in 10^7 and one bit in 10^{11} bits (Franta and Chlamtac, 1981: 21-22). The lowest possible error rate is desirable for the real-time application. Furthermore, Ethernet uses baseband signaling to place digital signals directly on the coaxial

cable, with the capability of transmission over 1500 meters. This is adequate for the use of Ethernet in the FCDL, as the length of cable required for the entire facility is 240 meters. Finally, tapping into the cable-based Ethernet network is as easy as connecting and disconnecting N-series connectors on a transceiver device, and this generally does not disturb other traffic on the network. Thus, if a simulator cab on the network is down for maintenance, it can be separated from the network with little trouble.

The class of elements that can be interconnected via the Ethernet network depends only on the availability of a Network Interface Board (NIB) for each element. The devices which FCDL personnel intend to use can all communicate via a parallel port, so the NIBs all must have a parallel port and some control lines for handshaking. Commercial NIBs are available, but were designed for particular processors, or cost more than one designed in-house. Chapter V discusses the functional specification for an NIB designed in-house, using state-of-the-art Ethernet controller and serial input/output chips, which will only limit the type of element to those with parallel ports.

Reliability of transmission is a must for the real-time man-in-the-loop flight simulations to be conducted. Ethernet is one of the more reliable types of networks for several reasons. First, it has distributed control, thereby not allowing a network failure from the failure of a single node. Secondly, the bus topology of Ethernet has the potential of being more reliable than a ring network in that for the latter each NIB must regenerate each message to pass it on, whereas in bus systems all NIBs except the one transmitting the message remain

passive, so regeneration is not necessary and the chance of an error being committed is lessened (Franta and Chlamtac, 1981: 13, 45). Lastly, Ethernet's use of cyclic redundancy checks (CRC), further enhances reliability (Franta and Chlamtac, 1981: 35, 45).

From the above discussion, Ethernet appeared to be appropriate. The question of Ethernet's capability to functionally perform the real-time task is very important, and is addressed in Chapter IV. The last consideration besides performance was the availability of hardware to implement a working network. The primary hardware needed were the chips implementing the Ethernet specification. Several chip sets were advertised as meeting the need, and they are discussed below.

Selection of Ethernet Controller Chips

As of late 1982, six groups were identified as possible sources for network controller chips implementing the Ethernet specification Version 1.0 (Digital Equipment Corporation and others, 1980). These are listed in Table II below (Burskyin, 1982: 80).

Of the possibilities presented in Table II, only the chip sets manufactured by Advanced Micro Devices (AMD) and Intel were seriously considered, as they were the most likely to be in production by mid-1984 when they would be needed by the FCDL. Although Seeq-3Com actually had chips already in production, they were less than desirable as they only implemented the basic interface control functions of collision handling, address generation, CRC generation and checking, and serialization and deserialization. By waiting for the more complex chips manufactured by Intel and AMD, capabilities such as on-chip memory management and, in the

TABLE II

Sources for Ethernet Chips

Original Developers	Availability of Samples
AMD-Mostek	Late 1983
Ungermann-Bass-Fujitsu	Late 1982
Intel-DEC-Xerox	Late 1982
National Semiconductor	Late 1983
Rockwell Int'l. Microelectronics Division	Late 1982
Seeq-3Com	Mid 1982

case of the Intel controller chip, diagnostic routines and error counters could be obtained. These latter functions, although not necessary based on the requirements stated in Chapter II, would make the implementer's task much easier by providing diagnostic routines that would otherwise have to be developed in-house. Furthermore, if it were necessary in some extreme situation, the Intel chips can be reconfigured to have a faster response time than Ethernet by reducing the number of overhead bits per transmission, and reducing the backoff time per collision (Intel Corporation, 1982: 4-7). Thus, the Intel chip set, consisting of the 82586 Local Communications Controller and the 82501 Serial I/O chip, was the most desirable for its capabilities, configurability, and availability. Availability was the weakest asset of this chip set, however, for although Intel originally advertised availability of

samples in late 1982, samples of the 82586 were not obtained until early 1984.

Selection of Transceiver

The transceiver is the interface between the NIB and the coaxial cable. It provides ground isolation between the coaxial cable and the transceiver cable connection to the NIB, level conversion between the signal levels on the transceiver cable and the coaxial cable, and a high impedance connection to the coaxial cable.

In that Ethernet transceivers are standardized in their function, only limited research was done to survey those available on the market. The only requirement was that it meet Ethernet specifications. A transceiver manufactured by 3Com Corporation was first brought to the FCDL's attention and, as it was available immediately and was equipped to easily connect to the proposed network, it was selected.

Selection of Coaxial Cable

The coaxial cable selected is manufactured by Belden (Belden Corporation, 1983: 64-65), and meets the Ethernet Specification 1.0 (Digital Equipment Corporation and others, 1980: Section 7.3). A length of 240 meters was determined adequate to span the FCDL and allow controllers to tap into the network. Figure 4 indicates a proposed layout.

Cables meeting the Ethernet specification have annular rings marked on the cable jacket along their entire length at 2.5 meter intervals (±5 centimeters). The rings indicate where a transceiver tap can be made to add a node to the network. The relative spacing between

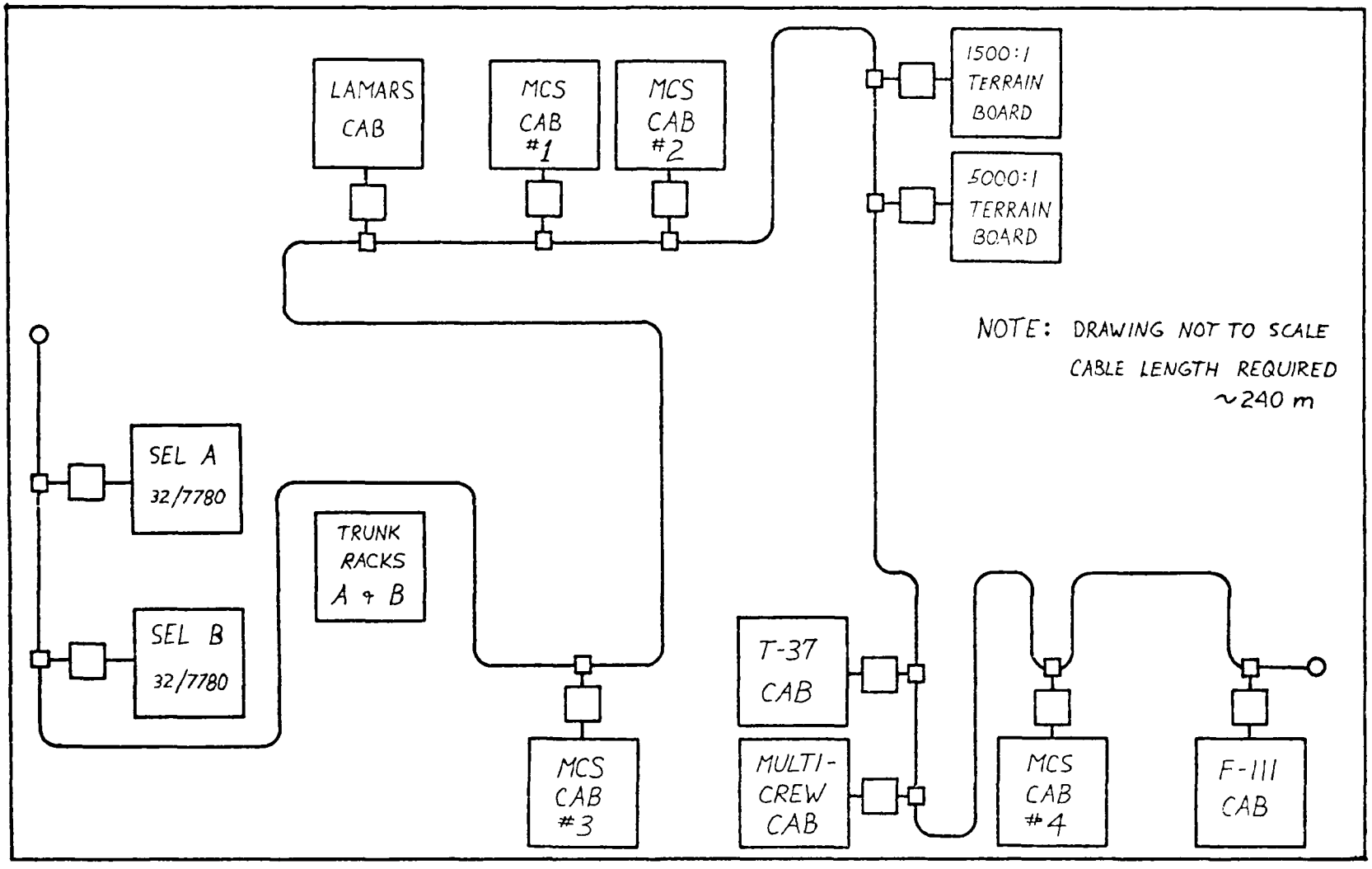


Figure 4. Proposed Ethernet Cable Layout

taps, and not the actual position on the cable, is the important point. It is necessary to minimize signal reflections due to the non-infinite bridging impedance of transceivers and their associated connections, and the controlled relative spacing ensures that the reflections do not add in phase to a significant degree (Digital Equipment Corporation and others, 1980: Section 7.6.2).

Selection of Transceiver Cable

The transceiver cable is a shielded twisted pair cable (four pairs) which connects the NIB to the transceiver. It can be up to fifty meters long and mates to the 3Com transceiver's 15-pin D-series male conector with an equivalent female connector. This cable was also procured from Belden (Belden Corporation, 1983: 64-65).

Expected Cost per Node

Assuming the LAN designed for the FCDL will eventually have 12 nodes, an average cost for connection of a node to the network can be estimated from the prices paid by the FCDL for some of the hardware, and from cost approximations for additional logic necessary for the NIB designed in-house. Appendix A summaries the relevant data.

Summary

In this chapter, a LAN architecture, Ethernet, has been shown to meet at least certain operational requirements necessary to transmit data for a real-time simulation environment, such as distributed control and reliability. The hardware to implement Ethernet was selected based on compatibility with the Ethernet specification and availability. These requirements are secondary, however, to the actual performance of

the LAN in delivering information real-time. The chosen LAN must ensure that all data packets intended to be transmitted and received within one cycle time of a simulation are indeed transmitted and received. The failure to meet this performance requirement defeats the use of the LAN. Chapter IV explores the expected performance of Ethernet in the real-time environment of the FCDL.

M+N = 2, Each Node Computes Own Missile Models. This scenario is the same as the first, except that the SEL and the MASSCOMP each compute their own missile models.

The states of First Pass IC and IC are identical to those shown in Figures 5 and 6, and are not repeated. The OP state of Figure 8, however, differs from Figure 7 in that the states of the MCS missiles are now passed from the MASSCOMP to the SEL instead of vice versa. Note that the number of bytes passed for this case and the first case are the same.

M+N = 4, Each Node Computes Own Missile Models. This scenario is the same as the first two in terms of data communication, except that each node must pass information to three other nodes instead of just one. Figures 9, 10, and 11 display the states of First Pass IC, IC, and OP for this case, respectively.

Network Throughput for Fixed Packet Size. Examination of Figures 5 through 11 reveals the following:

(1) The most number of bytes any one node must transfer during a simulation cycle occurs for M+N = 2 with the SEL computing the missile models (Figure 7), when the SEL must transfer 460 bytes when all four missiles are flying.

(2) Both M+N = 2 cases require three data packets to be transmitted during a simulation cycle, each less than or equal to 460 bytes in length.

(3) The M+N = 4 case requires five data packets to be transmitted during a simulation cycle, each less than or equal to 412 bytes in length.

In order to simplify the analysis, and to allow room for the addition of data to be transmitted, let a standard packet size of 500 bytes be defined, such that any transmission will consist of 500 bytes of data, whether it be useful data or not. In this way, an upper bound

Note that, for First Pass IC and IC, the missiles are still attached to the aircraft, so their states are the same as the aircraft's states. The inertial velocities and angular rates of both aircraft are transmitted from the SEL in order to initialize completely their states when entering the OP state. For example, the IC for an airplane may be a high-G turn, in which case just position information would not suffice. Note also that the SEL is the "master" in that it sends the MASSCOMP the First Pass IC conditions. Finally, during First Pass IC, the MASSCOMP only transmits status information to the SEL to indicate whether or not it came on line satisfactorily.

After First Pass IC, the simulation enters the IC state and waits for the OP state. During this time, the simulation operator is waiting for each pilot to give an indication that he/she is ready. The state information of each aircraft is transmitted to the other in order for the graphics software in the MCS controller and the target projector software in the LAMARS controller to be used and to present a target to each pilot in IC. The missiles are assumed to be still on the aircraft, and the rates passed in First Pass IC are assumed to be stored in the memory of the MCS controller.

In the OP state, the SEL is computing the states of its own aircraft and missiles, as well as the states of the MCS missiles. The status bytes are used to indicate the firing of a missile, the "killing" of an opponent, and other flags the user may deem necessary. The status bytes will also be used to indicate node or network failure in order to stop the simulation in a controlled manner.

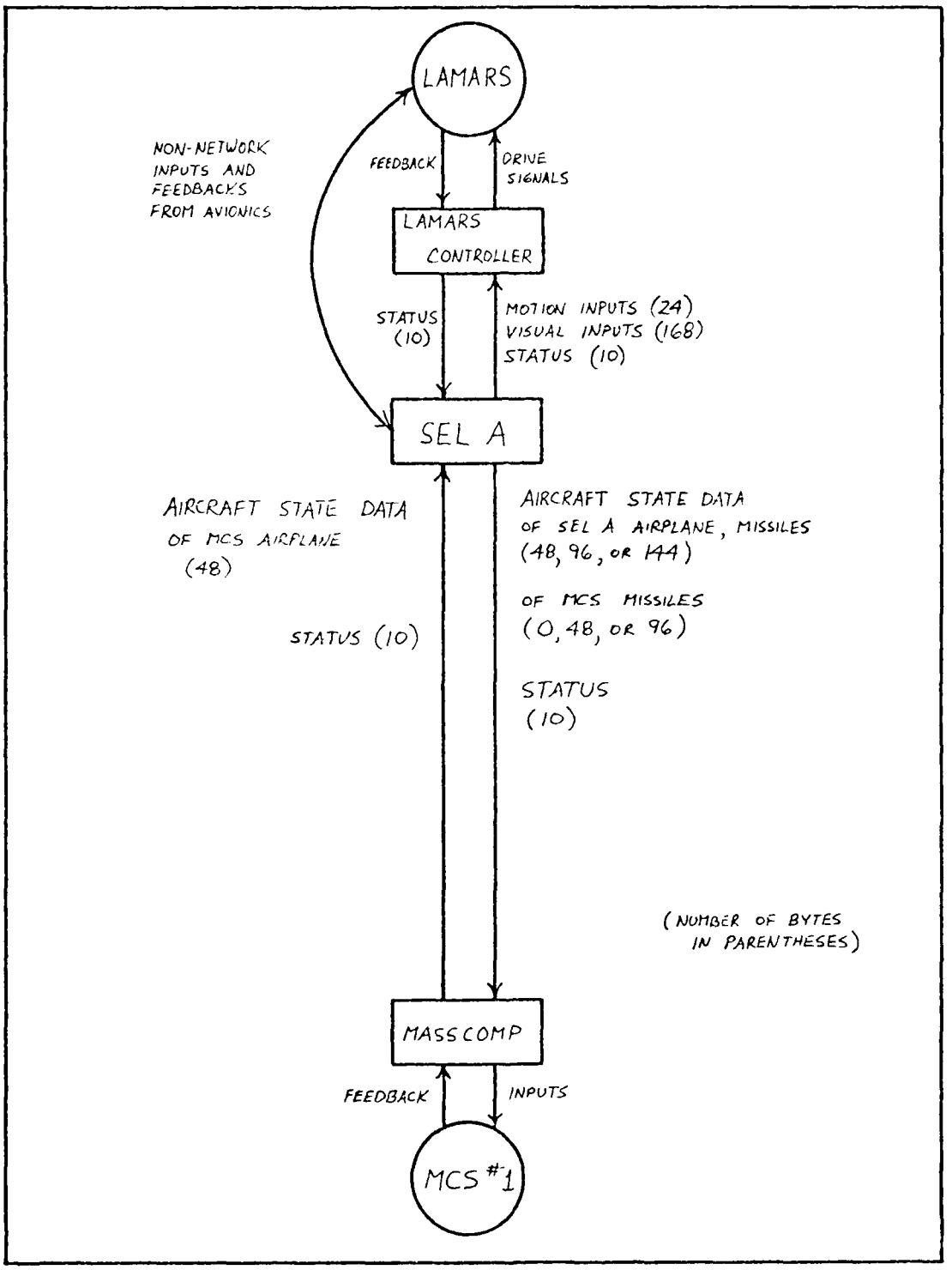


Figure 7. OP, M+N = 2, Missile Models in SEL A

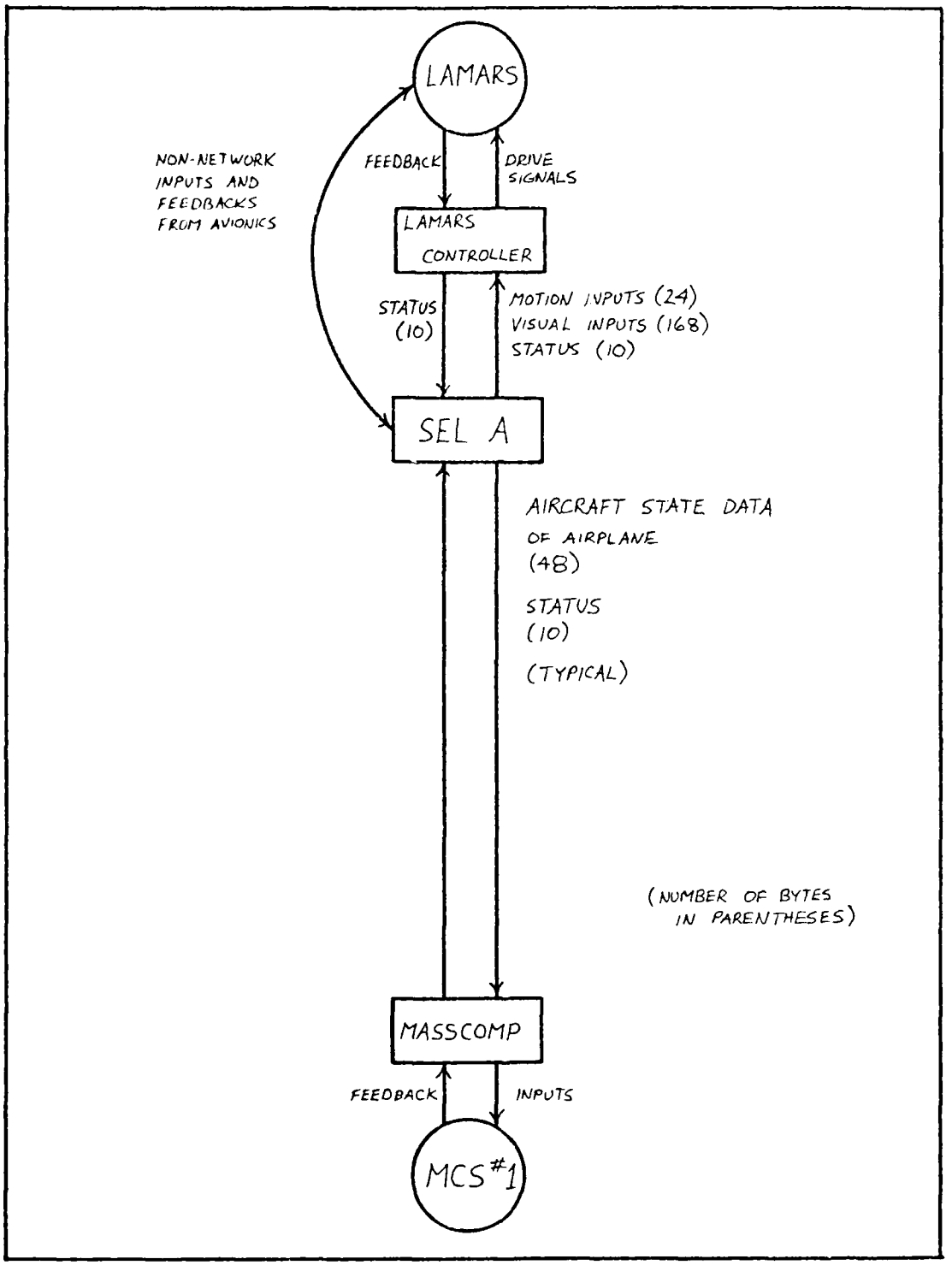


Figure 6. IC, M+N = 2, Missile Models in SEL A

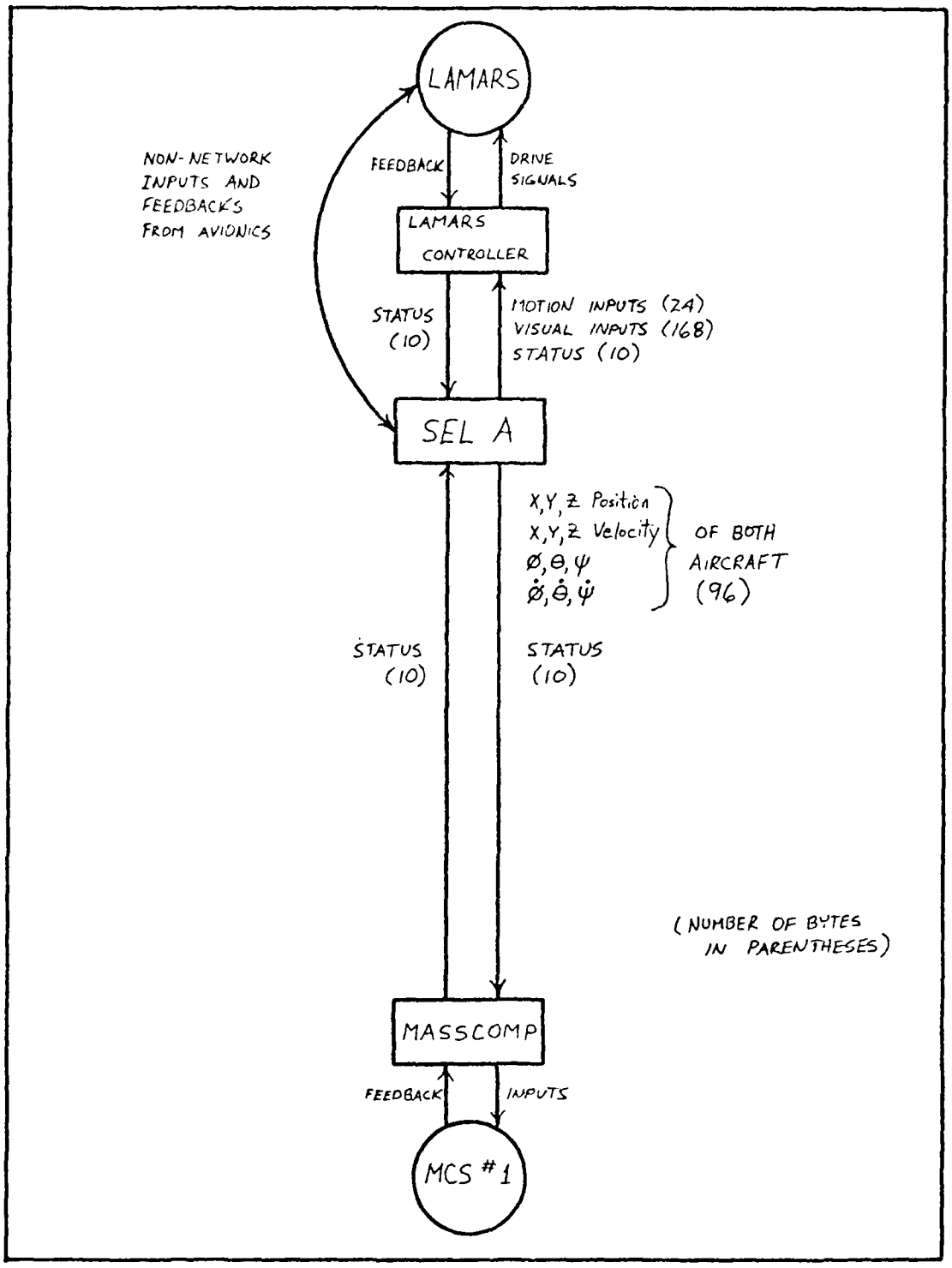


Figure 5. First Pass IC, M+N = 2, Missile Models in SEL A

HOLD is used to examine the status of a simulation at a chosen point in time, and essentially is obtained by holding fixed the state data of a simulation. Thus, it is similar to the OP state, but all data is being held constant.

With the possible states of a simulation and the content of data packets defined, a closer examination of the dynamics of data transmission can now be made.

Three Probable M-on-N Scenarios. As stated in Chapter II, the initial test of the FCDL network will be communication between the SEL and one peripheral computer, this being a one-on-one combat engagement. It is anticipated that each simulated aircraft will also have two missiles. The distributed computing problem referred to earlier is enough of a concern that both means of transferring the missile state data are presented below. As will be shown, the total number of bytes necessary to be transferred in either case is the same, so network throughput is not affected.

A follow-on test of the network will include an M-on-N simulation where M+N = 4. An example of the data communications necessary for this scenario is also presented below.

M+N = 2, Missile Models in SEL A. In this scenario, an aircraft simulated in SEL A is flying against one simulated in MCS #1. The pilot of the SEL A aircraft is using the LAMARS. All missile model calculations are performed in the SEL. Figures 5, 6, and 7 display the data communications necessary for the states of First Pass IC, IC, and OP, respectively.

improved hardware, a MASSCOMP can perform both the aero and missile models. Until that hardware is installed, though, the missile models will have to be computed in the SEL and the results transmitted over the network also. Examples of both computing distributions are presented later in this chapter for the case of an M-on-N simulation where M+N = 2 (two aircraft, each capable of firing two missiles). Another case is also presented for M+N = 4 (four aircraft, each capable of firing two missiles) where the peripheral controllers do compute their own missile models, because they should be capable of the additional load by the time this level of M-on-N is performed in the FCDL.

Simulation States - First Pass IC, IC, OP and HOLD. There are four states in which a simulation can reside: the first pass through that part of the software configuring the simulation for initial conditions, First Pass IC (also known as RESET); the remainder of the cycles through initial conditions, IC; the dynamic, real-time, operating portion of the simulation, OP; and a state in which the user freezes the simulation at a particular moment in time, HOLD.

First Pass IC informs each node of its individual operating state, and informs each node of the state of other nodes on the network involved in the simulation.

IC primarily serves the purpose of allowing each node to signal the "coordinating" node (SEL A, for instance) that it is ready to go to OP.

OP is the real-time operating state, during which all data packets must be transmitted and received within a simulation cycle time.

TABLE IV

Possible Transmissions for Network Nodes

Node	Input	Output
SEL A:	1) Aircraft State Data from Other Aircraft in Simulation	1) SEL A Aircraft State Data
	2) Status Data	2) LAMARS Controller Data 3) Terrain Board Commands 4) Status Data
SEL B:	Same as SEL A	1) SEL B Aircraft State Data 2) Terrain Board Commands 3) Status Data
LAMARS Controller:	1) LAMARS Controller Data	1) Status Data
1500:1 TB Controller:	1) Terrain Board Commands	1) Status Data
5000:1 TB Controller:	Same as 1500:1	1) Same as 1500:1
F-111 Controller:	Same as SEL A	1) F-111 Aircraft State Data 2) Terrain Board Commands 3) F-111 Moton Base Commands 4) Status Data
Multi-Crew Controller:	Same as SEL A	Same as F-111 for Multi-Crew
T-37 Controller:	Same as SEL A	1) T-37 Aircraft State Data 2) Terrain Board Commands 3) Status Data
MCS #1 Controller:	Same as SEL A	1) MCS #1 Aircraft State Data 2) Status Data
Other MCS Controllers:	Same as MCS #1	Same as MCS #1

TABLE III

LAMARS Controller Data (Continued)

Operations Status Data	Bytes
User Defined Flags to Control Simulation:	10
Terrain Board Commands	
Pilot Offset from C.G. in X	4
Pilot Offset from C.G. in Z	4
Aircraft Pitch, Roll, Yaw Rates	12
Tangent of the Glide Slope	4
Runway Origin North and East	8
Altitude Above Sea Level	4
Heading Relative to North	4
Altitude Bias	4
Terrain Board Scale 1 and 2	8
First Pass	.125
On/Off	.125
Total	44.25

Proposed Computing Distribution for M-on-N Simulations. When the idea was first being discussed of distributing the computation workload of the SEL 32/7780 via a LAN, the intent was to have each peripheral controller compute not only the aero model of the peripheral's airplane, but also that of (up to two) missiles it may fire. In this way, only state data of another node's aircraft and missiles need to be transmitted to the peripheral controller in order to compute relative state data. However, after examining the capabilties of the MASSCOMP computers to be used as the peripheral controllers, there is some question as to whether they can do much more than the aero model in the allotted cycle time of 25 milliseconds. There is some confidence that, with

TABLE III

LAMARS Controller Data

Motion Systems

Symbol	Meaning	Bytes
p_A	Aircraft Roll Rate	4
q_A	Aircraft Pitch Rate	4
r_A	Aircraft Yaw Rate	4
$\dot{u}_{PO}$	Pilot Observed Longitudinal Specific Force	4
$\dot{v}_{PO}$	Pilot Observed Lateral Specific Force	4
$\dot{w}_{PO}$	Pilot Observed Normal Specific Force	4
	Total	24

Visual System (for air-to-air target)

ϕ_{A_o}	Aircraft A Initial Roll Position	4
θ_{A_o}	Aircraft A Initial Pitch Position	4
ψ_{A_o}	Aircraft A Initial Yaw Position	4
u_A	Aircraft A Longitudinal Velocity	4
v_A	Aircraft A Lateral Velocity	4
w_A	Aircraft A Normal Velocity	4
p_A	Aircraft A Roll Rate	4
q_A	Aircraft A Pitch Rate	4
r_A	Aircraft A Yaw Rate	4
$X_{A_o}, Y_{A_o}, Z_{A_o}$	Aircraft A Initial Position	12
$X_{B_o}, Y_{B_o}, Z_{B_o}$	Aircraft B Initial Position	12
u_B	Aircraft B Longitudinal Velocity	4
v_B	Aircraft B Lateral Velocity	4
w_B	Aircraft B Normal Velocity	4
X_{PA}, Y_{PA}, Z_{PA}	Aircraft A Pilot Position	12
B Matrix	Aircraft B Direction Cosine Matrix	36
$\sin,\cos(\phi_A, \theta_A, \psi_A)$	Sine and Cosine of Aircraft A Angular Positions	24
X_{AB}, Y_{AB}, Z_{AB}	Relative Position of B to A	12
$\dot{u}_A, \dot{v}_A, \dot{w}_A$	Linear Accelerations of Aircraft A	12
$\dot{p}_A, \dot{q}_A, \dot{r}_A$	Angular Accelerations of Aircraft A	12
	Total	168

Aircraft State Information Data

Airplane:	X_A, Y_A, Z_A	Airplane Inertial Position	12
	$DCOS_A$	Airplane Direction Cosine Matrix	36
Each Missile:	X_M, Y_M, Z_M	Missile Inertial Position	12
	$DCOS_M$	Missile Direction Cosine Matrix	36

The Operations Status Data is used for the control of a simulation, and indicates states such as IC, OP, Hold and Reset (these states are defined later in the chapter). User defined flags must be included to allow network notification of an aircraft or missile having been destroyed, or of the failure of one of the nodes on the network. Note that even the absence of a Status data packet will inform the rest of the nodes of an unusual situation. It is through the use of this Status data packet that the user ensures the integrity of the simulation.

The Terrain Board Commands are used to drive a terrain board system, and are transmitted by the simulator controller using a terrain board to provide visual cues. As with the LAMARS, only one simulator can use a terrain board at a time.

Note in Table III that 4 bytes have been allocated for the real parameters because a SEL 32/7780 word consists of 4 bytes. The total number of bytes required to transfer each of the groups of data described above are also included in Table III.

With the types of data to be transmitted defined, the sources and destinations of these packets can be determined.

Data Packet Sources and Destinations. Each node on the FCDL network will be capable of transmitting and receiving data packets, but not all will transmit and receive all four categories as defined above. For example, there is no need for a terrain board controller to transmit LAMARS controller data. Table IV summarizes the packets that each node is capable of transmitting and receiving. Although five simulator controllers are shown as being capable of transmitting terrain board commands, again note that normally only one controller can transmit to a terrain board during a given simulation.

real-time fidelity. Typically, the relationship between throughput and delay is such that the higher the throughput the greater the expected delay (Franta and Chlamtac, 1981: 59-68). Therefore, the goal is to determine if, for the expected throughput requirement of the FCDL network, the expected delay of a transmission is acceptable within a high level of confidence. The first step is to determine the throughput requirements of the network, and this is discussed in the following section.

Throughput Requirements for the FCDL Network

To determine the throughput level at which the FCDL network must operate, it is necessary to study te actual data that must be transferred during a flight simulation.

Types of Data Transmitted. The content of the data packets to be transmitted over the FCDL network can be divided into four major categories: LAMARS Controller Data, Aircraft State Information Data, Operations Status Data, and Terrain Board Commands. Table III summarizes the information that each contain.

The LAMARS Controller Data, consisting of motion and visual system inputs, will be transmitted from only one node per simulation since only one aircraft can be simulated in LAMARS at a time.

The Aircraft State Information Data consists of the inertial positions and direction cosine matrix for each device to be tracked by other nodes on the network, where typical devices are aircraft and missiles. Thus, for a 1-on-1 combat engagement, each simulator controller would pass the state of its aircraft to the other in order to compute relative positions.

is known as the normalized propagation delay (Franta and Chlamtac, 1981: 58, and Stallings, 1984b: 28).

Of the four parameters commonly used, the relationship between throughput and delay is of primary concern in the analysis of the FCDL real-time network, and is discussed further in the following section.

Expected Delay and Real-Time Operation

Local area networks have typically been used more in an office environment for the transfer of files, electronic mail, data storage, and the like (Metcalfe and Boggs, 1976: 395-396; Shock and Hupp, 1980; Franta and Chlamtac, 1981: 13-21). These tasks are non-real-time in that, as long as the task is completed within a second, the user is satisfied with the network's performance. Minimizing delay can only be accomplished while satisfying high throughput requirements, as several hundred users may be using the network at one time. Thus, maintaining a high throughput capability is of primary concern in this type of use of a LAN.

In a real-time application of a LAN, however, expected delay is of primary concern. The FCDL performs flight simulations of high performance aircraft with a human operator in the control loop, and the typical cycle time of the digital computer performing the aerodynamic computations is 25 milliseconds (from personal experience with FCDL operations). Obviously, the proposed network must be able to complete a data transmission within this 25 milliseconds, or some node(s) on the network would have to use "old" data for the "new" cycle calculations. Every node on the network requiring it should have new data available every 25 milliseconds or less in order for the simulation to maintain its

IV. Analytical Performance Analysis of Network Hardware

This chapter presents the justification in terms of performance for using a CSMA/CD, or Carrier Sense Multiple Access/Collision Detection, (Ethernet-like) protocol for the real-time application intended for the FCDL. A discussion of the expected throughput requirements of the network results in two LAN protocols, CSMA/CD and token-passing, being the best suited to minimize data packet transmission delay. The advantages and disadvantages of each are examined, with CSMA/CD emerging as the more desirable. Finally, a prediction of the average delay a data packet may experience using the Ethernet protocol is determined.

Network Performance Measures

Network performance is usually given in terms of three parameters (Franta and Chlamtac, 1981: 58):

1) channel throughput S, the number of messages transmitted per unit of time;

2) channel capacity C, the maximum S value achievable with a given access protocol; and

3) the expected delay D, the time elapsing between the generation of a message and its successful transmission.

Another parameter used to characterize LAN performance is normalized propagation delay. If T is taken as the unit of time required for a transmitter to get an entire packet of information out onto the medium (such as a cable), and d is the channel propagation delay between the remotest pair of nodes on the network, then the quotient

$$a = (d/T) \qquad (2)$$

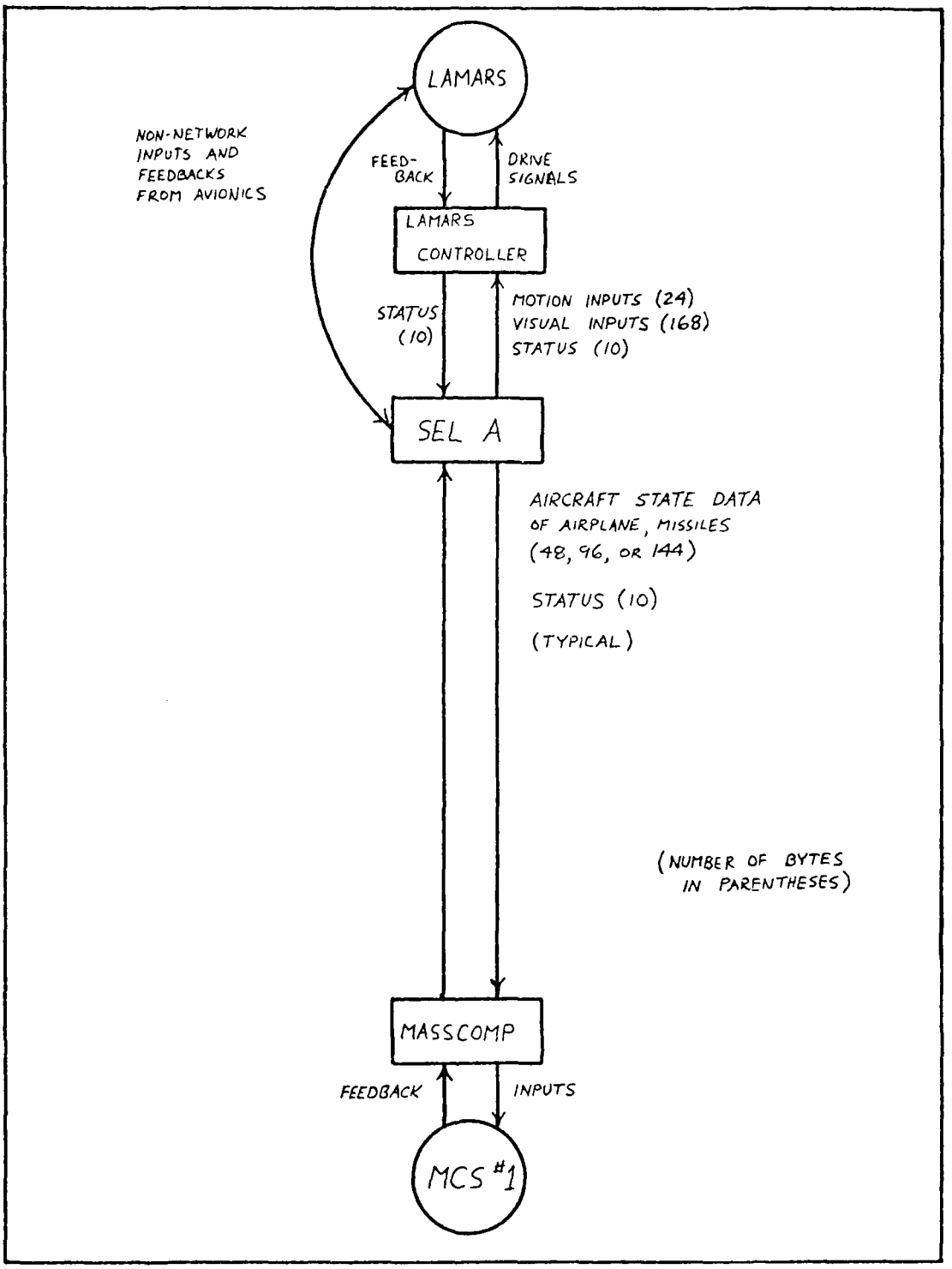


Figure 8. OP, M+N = 2, Nodes Compute Own Missile Models

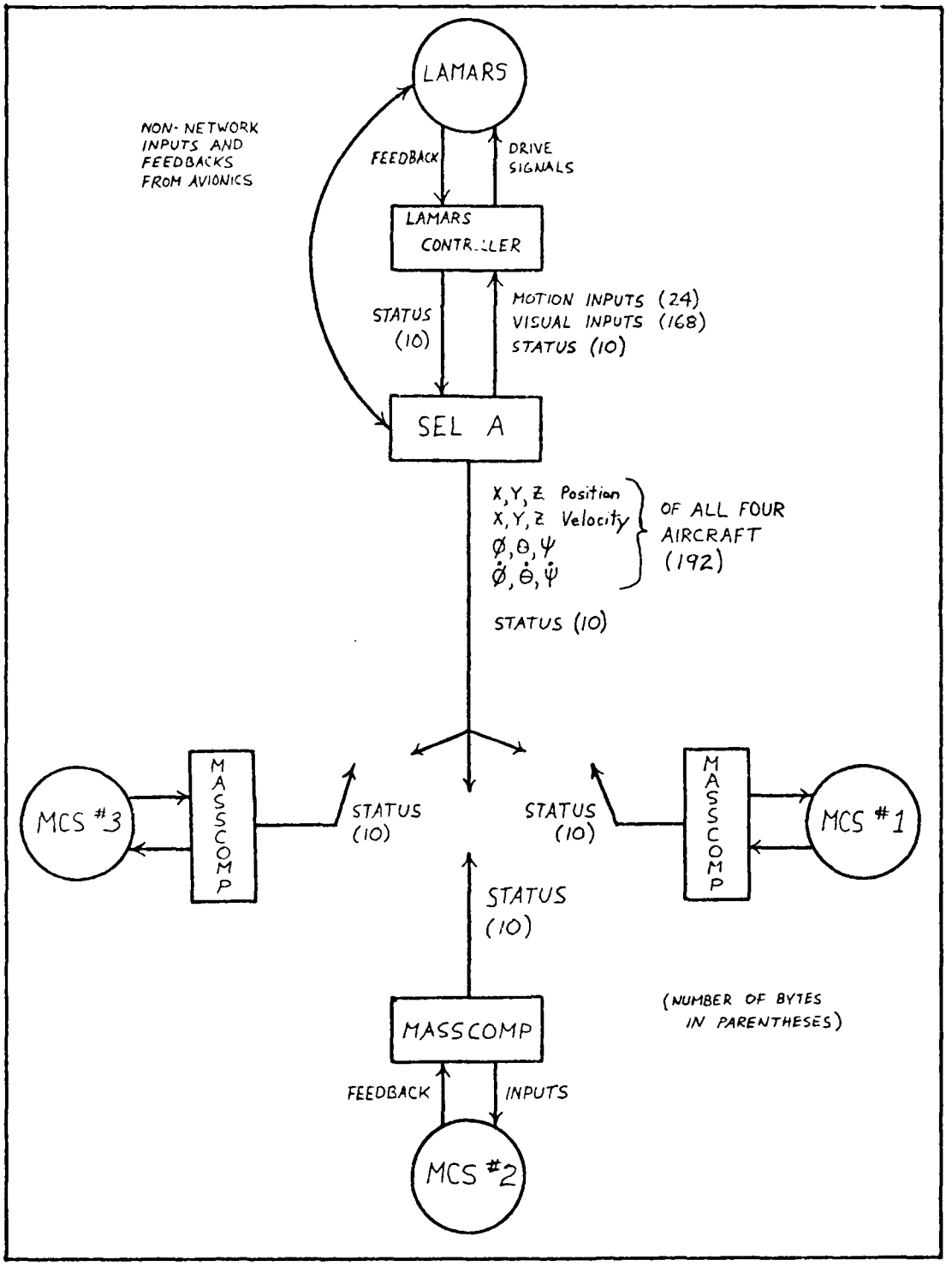


Figure 9. First Pass IC, M+N = 4, Nodes Compute Own Missile Models

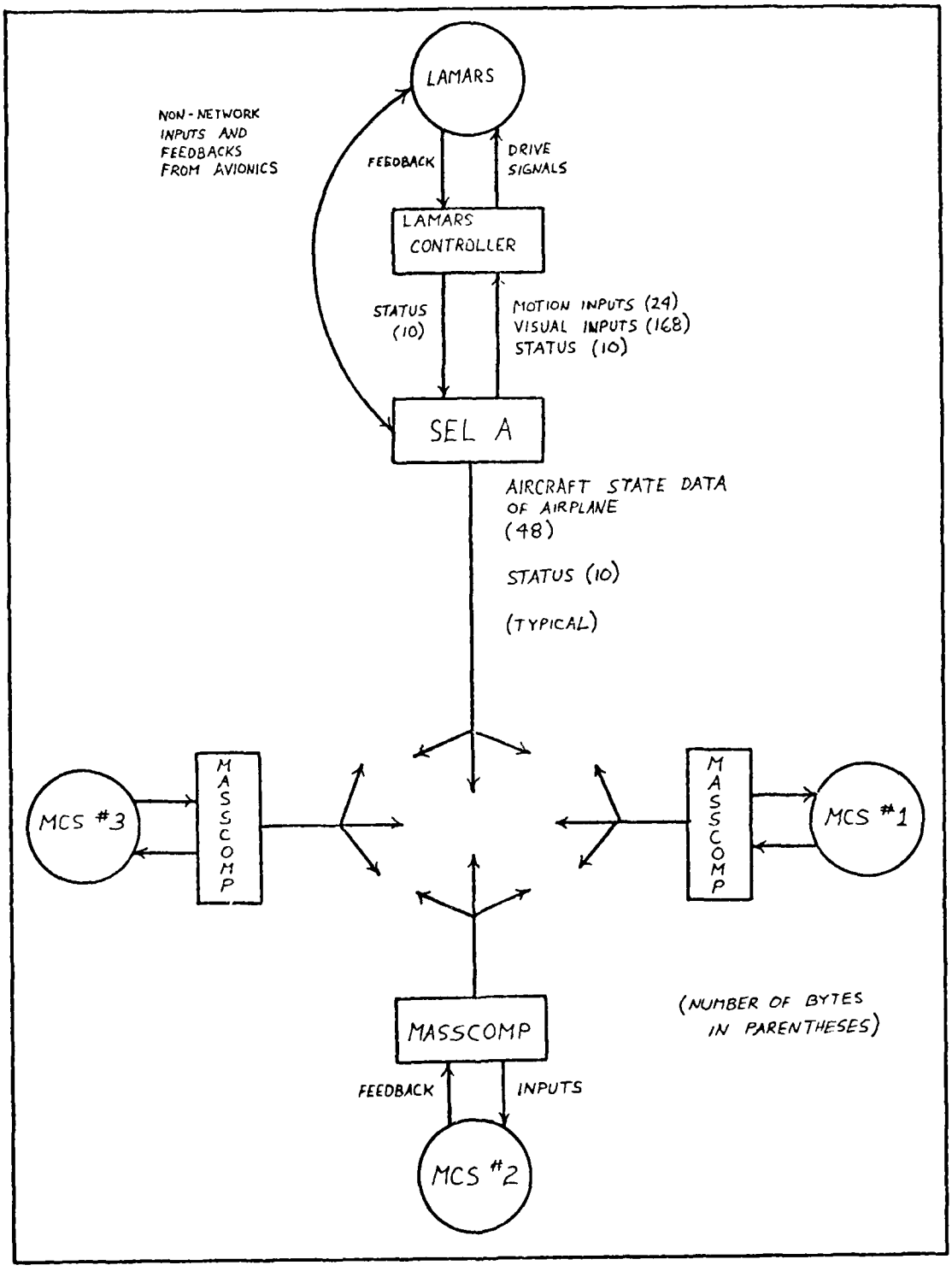


Figure 10. IC, M+N = 4, Nodes Compute Own Missile Models

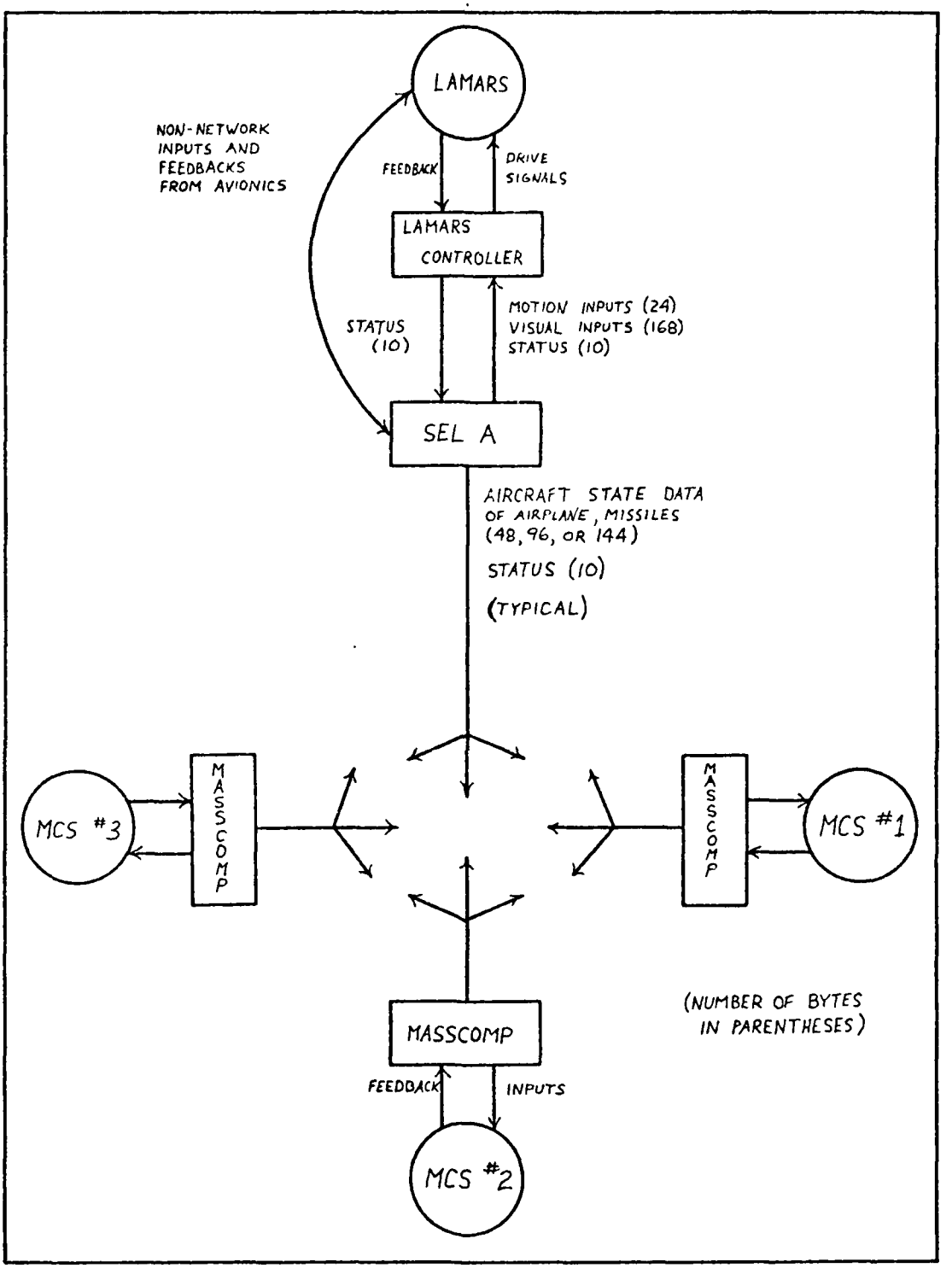


Figure 11. OP, M+N = 4, Nodes Compute Own Missile Models

on the amount of network throughput can be studied.

Let throughput be defined as the amount of useful data per unit time that is transferred over the network. Normalized throughput takes the latter quantity and divides it by the transmission rate of the network. Therefore,

Normalized throughput = (packets/second)*(bits/packet)/Trans. Rate (3)

The number of bits per packet is 4000, and the transmission rate is 10 Mb/s. The number of packets per second is determined by dividing the number of packets transmitted per simulation cycle by the simulation cycle time. For M+N = 2 the number of packets per second is 3/0.025, or 120 (each node on the network transmits one packet per cycle). For M+N = 4, the number of packets per second is 5/0.025, or 200. Thus,

$$\text{Normalized Throughput } (M+N = 2) = (120*4000)/10^6 = 0.048 \quad (4)$$

$$\text{Normalized Throughput } (M+N = 4) = (200*4000)/10^6 = 0.08 \quad (5)$$

Note that smaller packet sizes would yield even smaller throughput requirements on the network.

It is conceivable that multiple simulations could be using the network simultaneously. Referring to Figure 4 in Chapter I, it is observed that there are eight simulator cabs available: MCS #1, MCS #2, MCS #3, MCS #4, LAMARS, F-111, T-37, and Multi-Crew. Assuming this number will remain constant, and extrapolating the results from above, an M+N = 8 simulation could be performed and, assuming each simulator

cab controller could compute its own missile model, the throughput required would be 0.08 x 2, or 0.16. However, the "worst case" throughput requirement would be for four M+N = 2 simulations occurring simultaneously. In this case, the normalized throughput would be 0.048 x 4, or 0.192. In any event, then, the normalized throughput is predicted to be some value less than 0.2.

LANs Providing Low Delay for Low Throughput

Having determined the throughput range within which the FCDL network will operate, a literature survey was conducted to find available LANs with the smallest mean data packet transmission delay over that throughput range. The results of the survey revealed two protocols, CSMA/CD and token-passing, as having the best delay performance.

CSMA/CD is the protocol used for the Ethernet specification (Digital Equipment Corporation and others, 1980), and is being incorporated into the IEEE-802 standard for network protocols (Allan, 1982a). CSMA/CD minimizes the bandwidth wasted because of collisions by having nodes receive and transmit in parallel, so that a comparison failure between the transmitted and received messages signals a collision. When a comparison failure occurs, the node sensing collision jams the channel to guarantee that all currently transmitting nodes sense collision. After jamming, transmission is terminated ad reattempted after a random delay interval.

Token-passing accessing can be used on either a bus or a ring topology, and both methods are also being included in the IEEE-802 standard (Allan, 1982a). In the bus network, all network nodes receive

broadcast-type signals in any order, depending on the distance between the transmitting and receiving nodes and on the data transmission rate. In the ring network, individual nodes receive messages in sequence, and not necessarily related to the physical location of the nodes in the ring. For either topology, a token is passed from node to node. When a node receives the token, the node strips the token of the data intended for that node, and passes it to the next node. This procedure continues until the token returns to the original node. Token-passing accessing essentially provides that whichever node is holding the token has momentary control over the transmission medium.

The results of work by Bux (Bux, 1981) are most representative of the conditions under which the FCDL network will operate. Figure 12 shows Bux's results for the mean transfer delay-throughput characteristics for four protocols at 10 Mb/s assuming exponentially distributed packet lengths (Bux, 1981: 1469-1470). CSMA/CD is shown to have the lowest mean delay up to a normalized throughput of approximately 0.25, but is surpassed by the token-passing (ring) protocol at higher throughput rates. Bux's results for a constant packet length were similar (Bux, 1981: 1470). Several other efforts describing CSMA protocols can be found in the literature which also demonstrate characteristic good delay performance at low throughput (Lam, 1980; Tobagi and Hung, 1980; Vo-Dai, 1982; Stallings, 1984b; Stuck 1983; and Liu and others, 1982). The work by Lam, in fact, is used later in this chapter to predict the expected packet delay for the operating conditions of the FCDL network. Other literature relating to various aspects of performance for CSMA and its variants are also available to the interested reader (Blair and

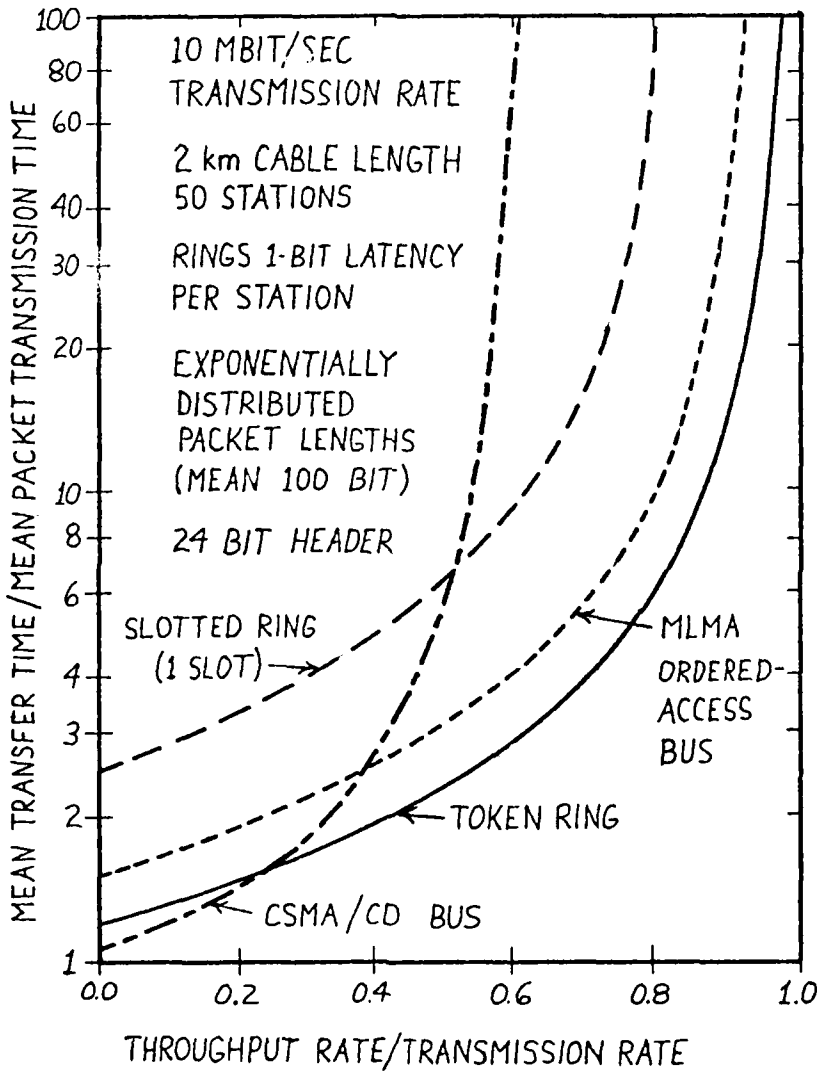


Figure 12. Transfer Delay - Throughput Characteristics of Four Subnetworks (Bux, 1981: 1470)

Shepherd, 1982; Shoch and Hupp, 1980; Shacham and Hunt, 1982; Metcalfe and Boggs, 1976; Arthurs and Stuck, 1982; and Tobagi, 1982).

Based on these results and the ease with which Ethernet can be implemented (from Chapter III), the CSMA/CD protocol is favored. Note, however, that Figure 12 is for mean delay, meaning that some packets in both protocols may be delayed more and some less. The advantage that token-passing has in this regard is that an upper bound on the expected delay of a transmission is known since the time a token takes to travel the length of the medium is known. Collisions on a CSMA-CD network, on the other hand, leave the determination of delays as an exercise in statistics and probability. Therefore, before Ethernet can be chosen with certainty, the probability of packets being delayed beyond the time length of a simulation cycle must be examined. This topic is explored in the next section to assist in the estimate of the total average packet delay for the FCDL network.

Total Average Delay Prediction for Proposed FCDL Network

This section outlines a best guess as to the projected delay in getting a data packet successfully transmitted from a node on one end of the FCDL LAN cable to a node on the other end. Items to be addressed include: the inherent hardware delays in an Ethernet type of LAN; an estimate of the mean packet delay based on the operating conditions of the FCDL network; the probability that the simulation cycle time will be exceeded; an estimate of the protocol software overhead delay; and the overhead imposed by an intervening High Speed Device (HSD) interface on a Gould/SEL 32/7780.

Physical Channel Delay. Table V summarizes the delays to be expected for an Ethernet LAN using devices meeting the Ethernet Specification (Digital Equipment Corporation and others, 1980: Section 7.1.5 and Table 6A-1) and the distances anticipated for the FCDL.

Therefore, the worst-case round-trip delay is estimated to be 9.5 microseconds. Note that the forward and return paths do take different amounts of time. This is because in one direction it is carrier sense which is being propagated through the channel, in the return direction it is collision detect which is being propagated, and the two signals have different propagation delays. For the purposes of this analysis, however, the one-way delay can be estimated by halving the round-trip estimate. The one-way delay, d, is then 9.5/2, or 4.75 microseconds.

Ideal Data Packet Transmission Time. As with all communication protocols, Ethernet adds overhead bits to the raw data to provide a preamble, addressing information, data typing, and frame checks. The total packet to be used by the FCDL will consist of the following (assuming the 500 byte data size):

Preamble	8 bytes
Destination Address Field	6 bytes
Source Address Field	6 bytes
Type Field	2 bytes
Data Field	500 bytes
Frame Check Sequence Field	4 bytes
Total	526 bytes

Ethernet also imposes a 9.6 microsecond idle time (interframe spacing) between transmitted frames to make sure one data packet fully propagates along the cable before another one begins (Digital Equipment Corporation and others, 1980: Section 6.3.2). The transmission time then becomes

TABLE V

Breakdown of Physical Channel Delay

Element	Unit Steady-State Delay (micro-seconds, unless noted	Unit Start-Up Delay (micro-seconds,	# Units Forward/ Return (units as noted)		Total Delay (micro-seconds)
Encoder	0.1	0	1	1	0.2
Tranceiver Cable	5.13 nS/m	0	50 m	50 m	0.513
Tranceiver (Transmit Path)	0.5	0.2	1	1	1.0
Transceiver (Receiver Path)	0.5	0.5	1	0	0.7
Transceiver (Collision Path)	0	0.5	0	1	0.5
Coax Cable	4.33 nS/m	0	240 m	240 m	2.078
Decoder	0.1	0.8	1	0	0.9
Carrier Sense	0	0.2	2	0	0.4
Collision Detect	0	0.2	0	2	0.4
Signal Rise Time (to 70% in 500 meters)	0	0.1	1	0	0.1
Signal Rise Time (50 - 94% in 500 meters)	0	2.7	0	1	2.7
			Round Trip Delay =		9.491

$$T = (526 \text{ bytes} \times (8 \text{ bits/byte}) \times (1 \text{ sec}/10*10^{6} \text{ bits})) + 9.6*10^{-6} \text{ sec} = 430.4 \text{ microseconds} \quad (6)$$

Note that this does not take into account channel assignment delays caused by collision resolution and deference. Rather, this is the mean transfer time of 500 bytes of data in the ideal Ethernet network of no collisions, and is included here for comparison with the more realistic transmission time, discussed below.

Mean Data Packet Transmission Time. Bux's results shown in Figure 12 were actually based on work performed by Lam (Lam, 1980). Lam developed a model of the CSMA/CD protocol and derived expression for the mean packet delay and a probability expression for channel assignment delay (Lam, 1980: 27). The expression for mean packet delay, discussed in Appendix B, is used to generate the data of Figure 13. The characteristics of the FCDL network were used in evaluating the expression. The three packet sizes of 1000, 2000, and 4000 bits (125, 250, and 500 bytes) were selected to show how packet size affects the mean delay. Note that, for a given throughput, the mean delay time of a smaller packet size is greater than that of a larger packet size. The reason for this is the ratio of overhead bits to usable data bits is higher for a smaller packet and, since throughput is based on the transfer of usable data bits, a higher percentage of time on the network is spent transferring overhead bits.

The most important data to be taken from Figure 13 is the mean delay for a 4000 bit (500 byte) packet in the throughput range fom 0 to 0.2, as this is the data packet size and throughput anticipated for use (from earlier discussion in this chapter). Recall that the maximum

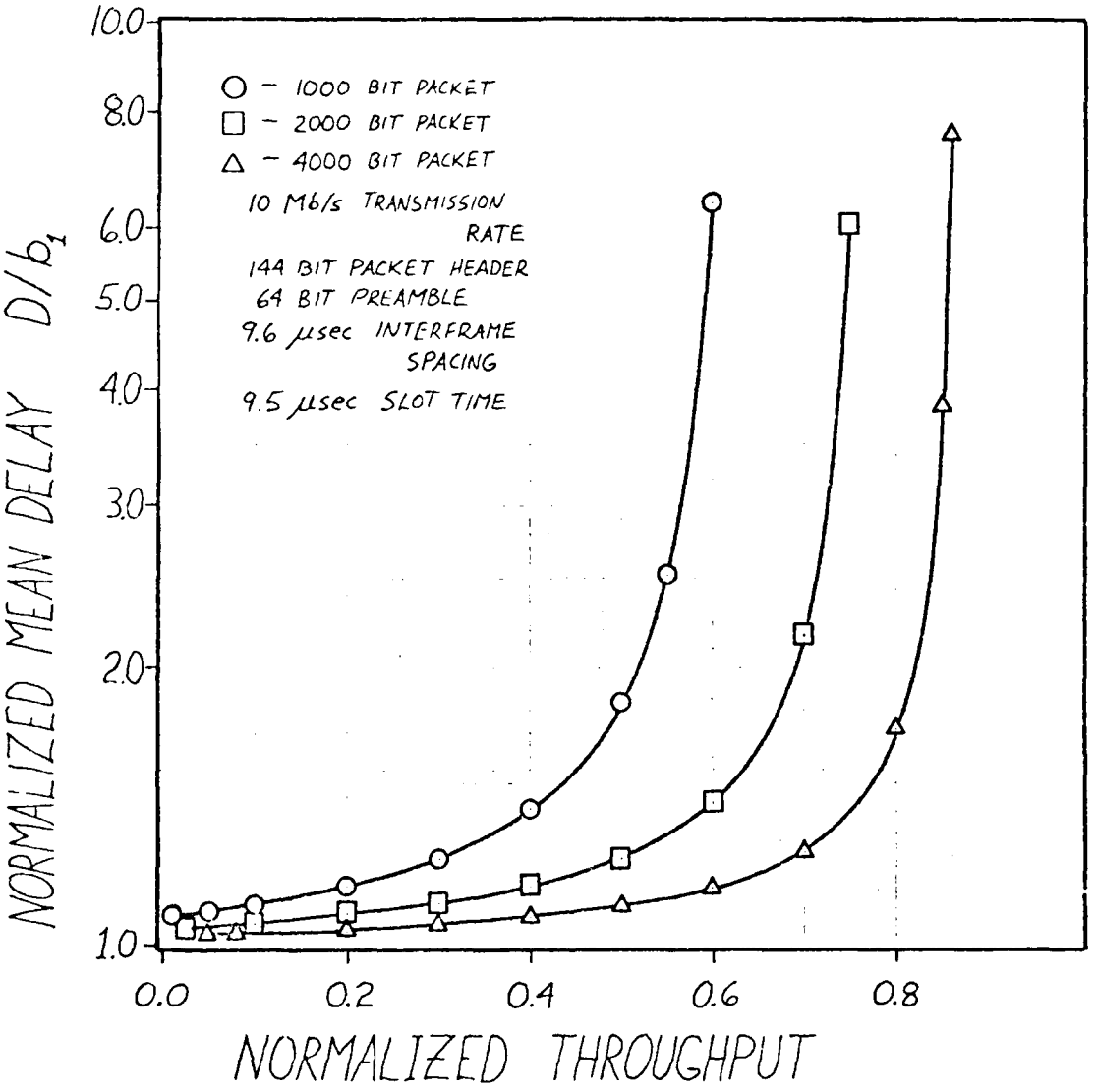


Figure 13. Transfer Delay - Throughput Characteristics of Proposed FCDL Network

throughput requirement is 0.192. At this throughput and for a 500 byte data packet, Figure 13 shows the mean delay to be approximately 1.04 times the ideal packet transmission time of 430.4 microseconds, or 447 mircroseconds.

Probability of Exceeding Cycle Time. The critical function of the FCDL network is assuring that all data is transmitted over the hardware within a simulation cycle. In that the mean packet delay has been shown to be at most 4 percent greater than the minimum possible, it appears that the Ethernet protocol can perform the required task. However, as one last check, Lam's channel assignment delay probability expression (discussed in Appendix B) was used to calculate the probability that a packet will be delayed longer than some amount of time such that, if on the average all packets were delayed this amount of time, the simulation cycle time would be exceeded.

Figure 14 presents the results of applying Lam's probability expression using the characteristics of the FCDL network. The vertical axis represents the probability of a packet being delayed longer than time $K_{max} * T$, where $K_{max} * T$ represents the maximum amount of time all packets could be delayed and still be transmitted before the end of the simulation cycle time. Again, the same three packet sizes used in Figure 13 are shown for comparison. Note that, for a given throughput, the probability of exceeding the cycle time for a smaller packet is greater than that for a larger packet. The reason for this is that more small packets can be transmitted at a given throughput than large packets, so there is more opportunity for collision- and deference-related delays.

VI. Conclusions and Recommendations

This chapter presents conclusions from the information of Chapters I though V, and contains a recommended course of action the Flight Contrc' Development Laboratory (FCDL) should take to complete the implementation of the network described.

Conclusions

The need for a real-time Local Area Network (LAN) in the FCDL flight simulation facility has resulted from the requirement to do M-on-N piloted aircraft engagements for the Manned Combat Station (MCS) project (where M+N is the total number of aircraft involved in th simulation, less missiles), and a desire to standardize the interfaces between computers.

Through the use of one interface board per computer, a computer connected to a LAN can communicate real-time with any other computer connected to the LAN via a standard protocol capable of transmiting data at a rate fast enough to be considered real-time. Additional computers may easily be added to the network as long as they conform to the real-time protocol. The use of such a LAN can allow the workload of the SEL 32/7780's during an M-on-N simulation to be distributed to peripheral processors, which is the intent of the MCS project. Another advantage of a real-time LAN is the elimination of the FCDL trunking stations for signal transmission.

To be connected to the network, a simulator cab must first have some kind of intelligent controller capable of communicating through a

be taken into consideration when using the HSD on the SEL have also been discussed.

The final chapter will summarize the results of this report, and recommend a course of action for the continued development of the FCDL network.

32/7780 to effectively transfer and receive data real-time to and from the network through the HSD, it must be able to accomplish these tasks during the real-time loop within the simulation cycle time. The SEL 32/7780 Input/Output Control System (IOCS), however, does not allow peripheral I/O to occur during the real-time loop. Therefore, a special IOCS must be written to "fool" the SEL 32/7780 into performing the HSD I/O. An IOCS of this nature already exists in the FCDL to accomplish the real-time transfer of data between the SEL 32/7780 and the Electronics Associates Incorporated (EAI) Pacer 100 digital computers. This IOCS enables the DDI (Digital-to-Digital Interface) in Figure 1 to operate correctly. It is possible that the user can use this DDI IOCS as a guide in the development of an HSD IOCS. This task is left for future study.

Until the HSD IOCS is written and implemented, the user can attempt to use the generic HSD data handler for a simulation as long as all data transfers occur during the non-real-time portion of the simulation cycle. The use and operation of an HSD is described in the HSD technical manual published by SEL (Gould, 1980). This task is also left for future study.

Summary

A software design for the control of the NIB by a host controller has been presented and discussed in this chapter and Appendix D. With the aid of the technical manual describing the details of the Intel 82586 Local Communications Controller (Intel, 1983), the user of the FCDL network may implement in software the design presented. Factors to

cannot recover, resetting the 82586 and attempting a restart is the only alternative. Note that, in either case, the node's absence from the real-time simulation will be detected by the absence of transmissions (status data) from the node. The other node(s) on the network can then decide whether or not to continue with the simulation without the defective node. The defective node does not dangerously impact the simulation because, upon detection of an error, the host ceases transmissions and receptions and enters a hold state. Faulty data will not enter the network, and the defective node will not receive new data. Therefore, a transition to either RDAVS or RESET from DIAGNOSTICS #2 is possible without causing harm to equipment or personnel.

Appendix D contains the formal software design describing the activities the host must execute for the NIB during each of the states described above. The user may use this design and the state diagram in Figure 15 as a guide in the development of software code in the language of the host.

SEL 32/7780 HSD Considerations

As mentioned in Chapter I, the interface between the SEL 32/7780 digital computer and the NIB requires the use of an HSD. This requirement adds some complication to the use of a SEL on the network that was not foreseen when the idea of the network was first conceived.

The original intent for implementing the HSD for use on the network was to use a generic HSD data handler written and supplied by SEL. This data handler would allow the HSD to be treated like any other input/output device connected to the SEL 32/7780. This latter characteristic, however, is where the problem lies. In order for the SEL

in NIB memory, and then is transmitted out onto the network. The status of the transmission is checked for errors by the host and, if no errors are found, the procedure is finished. Should an error be found, a transition is made to DIAGNOSTICS #2 as in the case of RDAVS.

The state RDAVS has priority over TDAVS since there is only one chance to collect the data from the network, whereas the host can retransmit a packet if necessary. Therefore, it is possible that the state TDAVS can be interrupted by the state RDAVS when a packet arrives at the NIB. If this occurs, the host must either save the condition it was in at the time of the interruption (such as the location and number of bytes left to be transferred to NIB memory, etc.), or it must restart the transfer. This situation is depicted by the transitions between RDAVS and TDAVS shown in Figure 15. The host may transfer data to the NIB only when the NIB is not receiving data from the network. Furthermore, if the host is transferring data to the NIB and a packet on the network needs to be received, control returns to RDAVS immediately until the packet is received. Control can then transition back to TDAVS to complete the transfer.

The state DIAGNOSTICS #2 is reached from states DIAGNOSTICS #1, RDAVS, and TDAVS when errors occur that cannot be resolved otherwise. DIAGNOSTICS #2 consists of the examination of the status word of the SCB, the status word of the command last executed by the 82586, and the execution of the 82586 Dump Status command in an attempt to locate the cause of the error(s). If the NIB can recover from the error (that is, if the host can resolve the error situation in the NIB), a transition can be made to RDAVS for re-entering real-time operation. If the NIB

exited following completion of these functions, and the transition is into the state DIAGNOSTICS #1.

Prior to entering the real-time states of data transmissions and receptions, some diagnostic functions should be performed of which the 82586 is capable. DIAGNOSTICS #1 is the state in which the following commands are performed: Diagnose, which tests the operation of the 82586 internal timers; Internal Loopback, which tests the transmission and reception capabilities of the 82586 internal to the chip; External Loopback, which tests the transmission and reception capabilities of the 82586 via the network transceiver; and Time Domain Reflectometer (TDR), which examines the network cable for discontinuities. If one of these tests is not passed and the cause of the failure cannot be determined, a transition is made to state DIAGNOSTICS #2. If all the tests pass, and the user desires to start the real-time simulation, a transition is made to state Receive Data And Verify Status (RDAVS).

Up to this point, all the states for initializing, configuring, and testing for the correct operation of the 82586 have been non-real-time. The transition to RDAVS signifies the transition to real-time operation. During real-time operation, the state of NIB transitions between RDAVS and Transmit Data And Verify Status (TDAVS). During the state RDAVS, data is stored into NIB memory as it comes in from the network. The status of the reception is checked for errors by the host and, if no errors are found, the data is transferred to the host. Should an error be found, a transition is made to the non-real-time state DIAGNOSTICS #2 for error resolution (and the NIB is essentially taken off-line). During the state TDAVS, data from the host is stored

state. This state is exited by the host applying channel attention to the NIB.

The state INITIALIZATION is reached by the host applying an input called "channel attention" to the NIB following RESET. The primary function of the initialization procedure performed during this state is to inform the NIB of the location of the 82586 System Control Block (SCB). The SCB is a memory block that is shared by the host and the 82586, and forms the communication link between the host and the 82586. The communications that occur include the issuing of commands to the 82586 by the host, and the reporting of command and operation status by the 82586. After the initialization sequence is completed, the 82586 is configured by default to the Ethernet specifications with broadcast addressing (the NIB will receive any packet on the network). Should the user decide to vary from these specifications for some reason (such as desiring a unique address for the NIB), the 82586 can be reconfigured by passing through the state CONFIGURE. Otherwise, the transition is to DIAGNOSTICS #1.

CONFIGURE is the state in which the NIB can be assigned a unique individual Address (IA), one or more Multicast Addresses (MA), and/or reconfigured to some non-Ethernet specification. This state is reached by the user who wants to perform one or more of these functions following INITIALIZATION. Typically, this state will be visited, as it will be desirable to assign a unique address to each node on the network so that a NIB won't be bothered with packets not intended for it. It is also desirable to assign a NIB with multicast addresses common to other NIBs so that one packet can be sent to multiple NIBs. The CONFIGURE state is

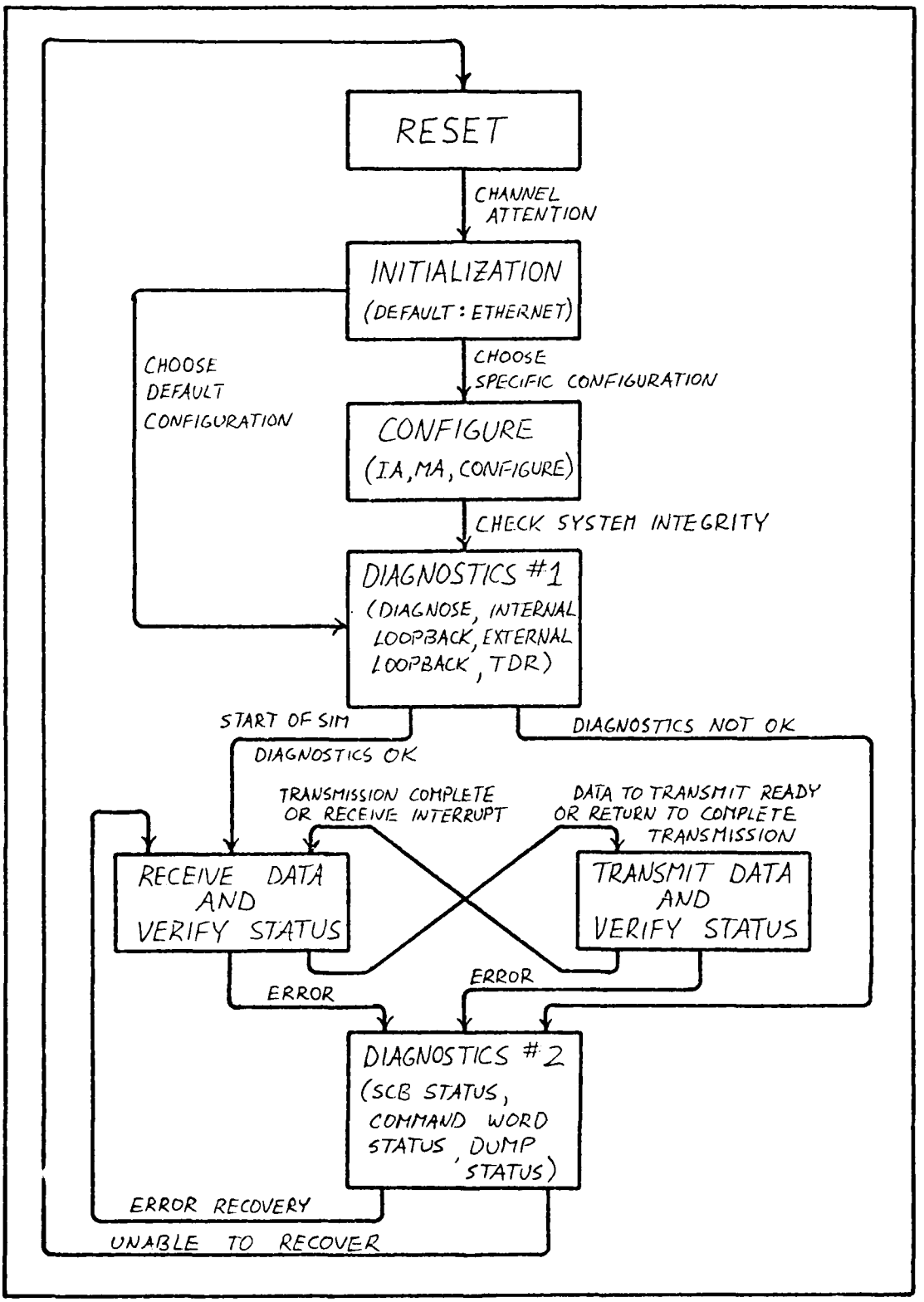


Figure 15. NIB State Diagram

as the details can be obtained in Intel's reference manual describing the chip (Intel Corporation, 1983). Only the details that pertain to the description of the software design will be included in this chapter.

The operation and capabilities of the 82586 provide the basis upon which both the NIB and the host controller software are designed. The NIB design was accomplished in-house, and its operation is discussed in Appendix C. The design of the host controller software is the topic of the next section.

Controlling the NIB for Real-Time Simulations

The control of the NIB by a host controller is basically performed by sending commands to the 82586 and prompting the 82586 to execute them. The types of operations that are necessary for communications on the network, and that the 82586 is capable of performing, include: initialization of the 82586; configuration of the 82586 (selecting the addresses by which the NIB will be identified); operational error diagnostics; data transmission; and data reception. Figure 15 presents a "state diagram" representation of the suggested software design for using the NIB for real-time simulation. The function of each state, and the actions causing transitions to another state, are discussed in the following paragraphs.

RESET is the state in which the NIB either originates upon powering up the NIB, or it is the state to which the NIB returns upon experiencing an unrecoverable error (the impact of an error on simulation operation is considered in the paragraph discussing state DIAGNOSTICS #2). All activity performed by the 82586 ceases while in this

V. A Software Design for Network Access

This chapter presents a software design for the control of the Network Interface Board (NIB) which allows communication over an Ethernet network for real-time simulation. Following a brief description of the Intel 82586 Local Communications Controller and the NIB, a diagram of the states in which the NIB can reside during a simulation is discussed. Factors that are important to the operation of a real-time simulation are noted. Diagrams of the activities a host controller must perform to place the NIB in these states are contained and discussed in Appendix D. Finally, implementation of the SEL 32/7780 HSD software necessary to perform the activities is discussed.

The Intel 82586 Local Communications Controller

The Intel 82586 chip is an intelligent, high performance local Communications Controller (LCC). It provides most of the functions normally associated with the data link and physical link layers of a local network architecture. Specifically, it performs frame boundary delineation, addressing, bit error detection, link management, and data modulation. Furthermore, the 82586 has diagnostic capability, in that it automatically gathers statistics on CRC errors, frame alignment errors, overrun errors, and frames lost because there was no resource to receive them. The status of all 82586 internal registers can be dumped to memory upon command to help in locating system faults, and the 82586 also has a time domain reflectometer that can be used to help locate network cable faults (Intel Corporation, 1983: 1). It is beyond the scope of this report to describe in detail the operation of the 82586,

Task	Delay (msec)
SEL I/O Software	1.0
SEL to NIB Transfer	0.621
NIB to NIB Transfer	0.447
SEL I/O Software	1.0
NIB to SEL Transfer	0.621
Total Delay	3.689

This value is well within the 25 milliseconds cycle time, and does not pose a threat to the fidelity of a real-time simulation.

Summary

The protocol used by the Ethernet specification, CSMA/CD, has been shown to have sufficient delay performance for the network application intended in the FCDL. Each packet sourced by a node on the network may take on the order of 4 milliseconds to reach its destination, but will only be on the network cable for approximately 0.5 milliseconds. As long as the throughput requirements placed on the network remain below 0.2 for 500 byte data packets, the transfer of all data packets within a siulation cycle is assured with a high level of confidence. The low utilization of the network is what assures few collisions, small delays and real-time operation.

The next chapter will present a software design for the control of the NIB using the Intel 82586 Local Communications Controller chip chosen in Chapter III. This software design must be implemented for each host controller desiring to communicate on the network.

initial configuration of the FCDL network will have a Gould/SEL 2/7780 communicating with a MASSCOMP microcomputer. The SEL will be the slowest device, as it must connect to a NIB through a Model 9132 High Speed Data (HSD) Interface, and its throughput is assumed to be lower than the MASSCOMP's I/O throughput. Therefore, the transfer of a packet through the HSD will be considered worst case.

According to the HSD Technical Manual (Gould, 1980), the throughput associated with the HSD cannot be explicity defined since it is highly dependent on the system configuration, the HSD bus priorities, the actual data rates of the devices attached to the HSD, the cable lengths, the number of HSD's in the system, and where in memory the data is being processed in relation to where the CPU is executing. Therefore, assumptions were made by the author in order to provide some guideline. Assuming that only memory Bus Controllers (MBC's) have a higher SEL bus priority than HSD's, and the cable length between the HSD and the device controller (NIB) is 20 feet to 50 feet long, then one of two HSD's may access the same memory module as the CPU and other I/O Devices at a maximum rate of 1.2 microseconds or 834 kilowords per second (KW/s) each (Gould, 1980: 3-114). If this rate is achieved, the 518 byte packets to be used (the preamble is affixed at the NIB) will require approximately 0.621 milliseconds to be transferred.

Packet Transfer Total Average Delay. The total estimated average time required to transfer a packet from the memory of a SEL computer to the memory of another SEL computer on the network (again, consider worst-case) would be the sum of the delays discussed above:

The most important point to be made from Figure 14 is that the probability of the cycle time being exceeded for the throughput range of 0 to 0.2 and 500 byte data packets is insignificant (less than 10^{-39}). Therefore, one can say with a high degree of confidence that the Ethernet protocol will perform in the real-time application in the FCDL.

With the average delays in the Ethernet hardware thus established, the remaining factors making up the total average delay in the transmission of data from one controller to another must be addressed.

Protocol Software Overhead Delay. The total delay in transmitting a packet not only arises from inherent network hardware delays as discussed in the last few sections but also from the higher-level software implementing the protocol. At this point in the analysis, only a best guess can be made, as the actual delay could only be effectively determined through measurement of a working system, and/or by summing the estimated execution times of each command in the software itself. As the software is not yet written, an estimate will have to suffice. As an estimate, based on discussions with personnel in the FCDL familiar with existing system software, assume the execution of the software used to transfer a data packet from the intelligent controller to the NIB takes on the order of 1.0 milliseconds.

Hardware Delay from Controller to NIB. The final portion of a data packet's delay in reaching a destination is in its transfer from the intelligent controller to the NIB. Just as it take time for the packet to get transmitted from the NIB onto the network, it takes a certain amount of time for the packet to go from the controller's memory to the NIB's memory prior to being placed on the network cable. The

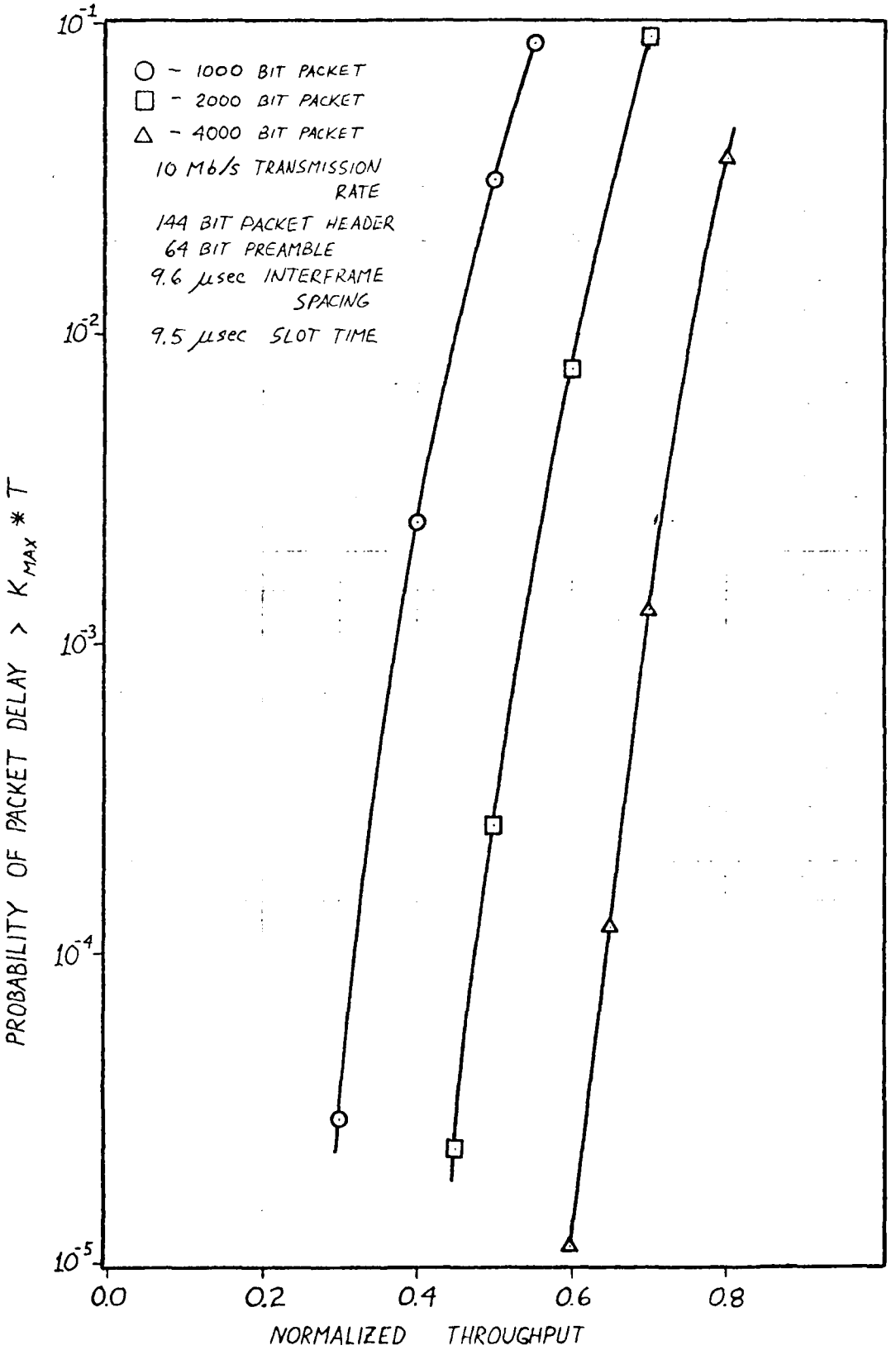


Figure 14. Packet Delay Probability - Throughput Characteristice of Proposed FCDL Network

parallel port. Secondly, a network interface board, or NIB, must exist to allow the physical connection to the network. One of the goals of the MCS project is to design and build this NIB. The SEL 32/7780's have the additional requirement of using a High Speed Device (HSD) interface with the NIB. Finally, a network protocol is required to perform communications quickly and efficiently.

The FCDL user requirements for a LAN are stated in terms of network protocol requirements hardware requirements, operational requirements and cost constraints. The two primary requirements are that the network have a data transfer rate of at least 10 Mb/s, and that it be low cost. A survey of available LAN's revealed that the Ethernet protocol was the least costly of those meeting the transfer rate requirement. Futhermore, its physical operating characteristics lend itself for use in the FCDL: digital transmission; a bus topology, which is resistant to single point failures; low bit error rate coaxial cable as the transmission medium; easy interconnection of devices on the network; and good reliability.

Based on its satisfactory physical operating characteristics, the hardware necessary to implement Ethernet was selected. The Intel 82586 Local Communications Controller (LCC) and 82501 Serial I/O chip set was selected for its capabilities in satisfying FCDL requirements, configurability (should the user decide to vary from the Ethernet protocol for some reason), and availability. Ethernet-compatible transceivers, coaxial cable, and transceiver cable were also selected.

A performance analysis of an Ethernet LAN using the hardware selected and the operating environment of the FCDL was conducted. The

performance measures of throughput and delay were of primary interest. The goal was to determine if, for the expected throughput requirement of the FCDL network, the expected delay of a transmission is acceptable within a high level of confidence. Examination of data packet transmissions necessary to conduct real-time flight simulations in the FCDL reveal that the maximum normalized throughput requirement is 0.192. Although both token-passing and Collision Sense Multiple Access/Collission Detect (CSMA/CD, or Ethernet) protocols are shown to have low mean packet delays for this level of throughput, CSMA/CD is lower than token-passing. Furthermore, it is shown that, for the throughput levels anticipated for the FCDL network, the chance of a data packet not being transferred within the cycle time of a flight simulation is negligible (less than 10^{-39}). Therefore, one can say with a high level of confidence that the Ethernet protocol will perform satisfactorily in the real-time application in the FCDL. Finally, the performance analysis shows that the total average delay of a packet transmitted by one SEL 32/7780 and received by another over the proposed network is on the order of 3.7 milliseconds. The transfer delay between a SEL and a MASSCOMP is anticipated to be even shorter. This amount of delay does not pose a threat to the fidelity of a real-time simulation having a typical cycle time of 25 milliseconds.

The selection of the Intel chip set, the in-house design and manufacture of a Network Inerface Board (NIB) using this chip, and the standard operating procedure of FCDL simulations allowed a software design and state diagram to be completed for the control of the NIB. The software design takes into account the diagnostics capability of the

Intel 82586 LCC to detect errors in the operation of both the chip and the network interface. Should a node experience an error during a simulation, the remaining nodes can detect its absence during the next simulation cycle by the lack of packets usually received from it. In this way these nodes can determine whether or not to continue the simulation. The node in error, upon the error detection, effectively goes into a hold state. All nodes can then be reset to try again. The user of the network may use this software design and the state diagram as a guide in the development of software code in the langugage of the host.

The SEL 32/7780's use of the HSD requires special consideration. A generic data handler developed by SEL will indeed allow I/O to occur, but the SEL I/O Control System (IOCS) does not allow it to occur during the real-time loop of a simulation. Therefore, a special IOCS must be written to "fool" the SEL into performing the HSD I/O. An IOCS of this nature already exists in the FCDL, and is known as the Digital-to-Digital Interface (DDI). It may be used as a guide in the development of a new HSD IOCS. In the meantime, the generic HSD data handler can be tested for use in a simulation as long as all data transfers occur during the non-real-time portion of the simulation cycle.

Recommendations

Before the proposed FCDL network can be used to communicate data between a SEL 32/7780 and a MASSCOMP, several interim steps are recommended to bring it to operational status.

First, the coding of DDI-equivalent NIB control software for the SEL 32/7780 HSD will require additional effort by the programmer as

compared to the NIB control software for a MASSCOMP. The MASSCOMP data handler will be easier to code as there is no HSD to include, and the data handler may be written in a language such as FORTRAN. Therefore, the FCDL could have an operational network between two MASSCOMPs relatively soon in the future. Any problems with transferring data real-time on the network can be examined and resolved by using this configuration. Therefore, it is recommended that the FCDL first write the NIB control software for the MASSCOMP.

Second, the initial attempt to connect the SEL to the network can be made by using the generic HSD data handler written and supplied by SEL. Although it cannot be used to transfer data during the real-time loop of a simulation cycle, there may be enough time in the non-real-time portion of the cycle to perform some transmissions. As the handler does exist, its implementation may not take too much effort. If it works, an interim solution to the problem of special DDI-equivalent NIB control software for the HSD is found. Any general problems with the use of the HSD on the real-time network can be resolved while implementing this data handler to ensure they don't arise when the special data handler is implemented. Therefore, it is recommended that the FCDL examine the use of the generic HSD data handler prior to writing the special data handler.

Finally, the DDI-equivalent NIB control software for the HSD should be written. The existence of the DDI data handler should help in the development of the HSD data handler. Once the special handler is completed, the network will be ready to perform real-time data transfers between the SEL and a MASSCOMP.

The order of this recommended approach is based on the time anticipated to be required to accomplish them, as well as on a logical progression towards bringing the network to full operation.

Appendix A: Expected Cost of Ethernet Connection Per Node

The expected cost per node of making a connection to an Ethernet network is totalled below, based on the prices quoted or estimated for the hardware described in Chapter III. Some of the items are divided by twelve to indicate the distributed cost over the anticipated maximum of twelve nodes on the network. Note that, less the cost of the wire-wrap board and wire, the NIB components cost about $380 of the $944 estimate. Note also that these prices are for the prototype NIB designed and built in-house at the FCDL, and do not include labor costs.

Ethernet Chip Set, 1 set per node	$180
Transceiver, 1 per node	$450
Coax Cable, $808/12 nodes	$ 67
Transceiver Cable, $96/12 nodes	$ 8
Terminators, $216/12 nodes	$ 18
Coax Cable Connectors, 2 per node	$ 6
Transceiver Cable Connectors, 3 per node	$ 15
Additional logic for NIB, per node	$200 (estimated)
Total	$944

Appendix B: Analytical Method for Predicting FCDL Ethernet Performance

In order to better justify the use of a CSMA/CD (Ethernet-like) protocol for the FCDL network, some quantitative measure of the expected message delay over the network was needed. A survey of literature revealed work by Lam (Lam, 1980), who modeled the CSMA/CD protocol and developed formulas for the mean message delay and mean channel assignment delay. A relationship for the probability of the channel assignment delay equalling an integer multiple of the slot time (twice the end-to-end propagation delay) is also stated. These formulas are applied using the characteristics of the FCDL network, and appropriate assumptions.

Mean Message Delay

The mean message delay (the time from arrival of a packet to the time it is successfully transmitted) is given by Lam (Lam, 1980: 27) as

$$D = \bar{X} + \frac{T}{S} + \frac{T}{2} - \frac{1-p_0}{2[B^*(\lambda)S-(1-p_0)]}\left(\frac{2}{\lambda} + ST - 3T\right)$$

$$+ \frac{\lambda[\bar{X^2} + 2\bar{X}(T/S) + T^2(1+2(1-S)/S^2)]}{2[1-\lambda(\bar{X}+T/S)]} \qquad (B.1)$$

where

$$\bar{x} = b_1 + \tau$$

b_1 = the mean of the packet transmission time Probability Density Function (PDF)

τ = the one-way propagation delay

T = the slot time of 2τ

S = the probability of a successful transmission

$P_j = \frac{(\lambda\tau)^j e^{-\lambda\tau}}{j!}$ = probability of j new arrivals in a time slot

$B^*(\lambda)$ = the Laplace transform of the packet transmission time PDF

λ = the arrival rate of messages to be ransmitted in packets/second.

$\overline{x^2} = b_2 + 2b_1\tau + \tau^2$

b_2 = the second moment of the packet transmission time PDF

For the purposes of this analysis, constant packet lengths are assumed. Therefore, b_1 can be defined as

$b_1 = ((L_P + L_H)/v)$ + Interframe Spacing

where

L_P = the length of the data in bits

L_H = the length of the data packet header and preamble in bits

= 208 for Ethernet

v = the network transmission rate in bits/second

= 10 Mb/s for Ethernet

The value of Interframe Spacing is 9.6 microseconds for Ethernet.

The one-way propagation delay τ is defined from Chapter IV as equal to 4.75 microseconds.

The slot time T is therefore equal to 9.5 microseconds. T is the maximum time, from the start of a transmission, required to detect a collision. Lam's derivation is based on the consideration that time on

the medium is organized into these slots, and it is convenient way of looking at activity on the network.

The probability S of a successful transmission during a contention period is assumed to be 1/e, which is the optimum slotted throughput rate for an infinite population model. Stallings (Stallings, 1984: 31-32) discusses the suitability of this assumption.

$B^*(\lambda)$ for the case of constant packet lengths is equal to one since the Laplace transform of a unit impulse function is one.

The value of b_2 is zero as it is the second moment of a constant value.

The value of $\overline{x^2}$ is assumed to be zero since the value of b_2 is zero, and the last two terms are on the order of 10^{-9} and 10^{-11} respectively.

Using equation (B.1) and the above assumptions, the normalized delay and throughput were calculated for packet sizes of 1000, 2000, and 4000 bits (125, 250, and 500 bytes) at different values of λ. The data is tabulated in Tables B.1, B.2, and B.3.

Channel Assignment Delay Probability

As Ethernet is probabilistic in the actual delay a packet will experience, some measure is needed of the probability that all packets will be transmitted within a simulation cycle. A formula stated by Lam (Lam, 1980: 27) gives the probability of the mean channel assignment delay equalling an integer multiple of the slot time:

$$Prob[d = KT] = \left[Q_0\left(1 - \frac{p_1}{1-p_0}\right) + \sum_{i=2}^{\infty} Q_i\right] S(1-S)^{K-1}, \quad K = 1, 2, \ldots \qquad \text{(B.2)}$$

where

$$Q_0 = \frac{1-\lambda(b_1+\tau+T/S)}{\lambda T\left(\frac{1}{1-p_0}-\frac{1}{B^*(\lambda)S}\right)}$$

and

$$Q_1 = \left(\frac{1}{B^*(\lambda)}-\frac{p_1}{1-p_0}\right)Q_0$$

Using equation (B.2), an expression can be derived for determining the probability that a packet will be delayed greater than some integer multiple of the slot time such that, if every packet in the cycle time is delayed that amount, the cycle time is exceeded.

First, an expressi. relating the time remaining in a simulation cycle to the arrival rate λ and the packet size must be stated. This expression is

$$\Delta = \frac{1-\lambda T_p}{\lambda} \qquad \text{(B.3)}$$

The "integer multiple of T" delay that each packet can incur before the simulation cycle is exceeded is then

$$K_{MAX} = INT\left[\frac{\Delta}{2\tau}\right] \qquad \text{(B.4)}$$

Then, the probability that a packet will be delayed greater than $K_{max}*T$ can be given, after some simplification, by:

$$Prob[d > K_{MAX} * T] = 1 - Q_0 - S(1-Q_0)\sum_{K=1}^{K_{MAX}}[(1-S)^{K-1}] \qquad \text{(B.5)}$$

Values of K_{max} and $Prob[d>K_{max}*T]$ were calculated for packet sizes of 1000, 2000, and 4000 bits (125, 250, and 500 bytes) at different values of λ. The data is tabulated in Tables B.1, B.2, and B.3.

TABLE B.1

Normalized Delay and Probability of Delay

for 125 Byte Data Packet

(Mean Transmission Time b_1 = 0.1304 msec)

λ (packets/sec)	D (msec)	K_{max}	D/b_1	Prob.	$\frac{\lambda * L_p}{v}$
120	0.1403	863	1.076	$< 10^{-39}$	0.012
200	0.1407	512	1.079	"	0.020
500	0.1419	196	1.088	"	0.050
1000	0.1444	91	1.107	"	0.100
2000	0.1511	38	1.159	$7.780*10^{-9}$	0.200
3000	0.1621	21	1.243	$2.939*10^{-5}$	0.300
4000	0.1834	12	1.406	$2.487*10^{-3}$	0.400
5000	0.2401	7	1.841	$3.153*10^{-2}$	0.500
5500	0.3284	5	2.518	$8.784*10^{-2}$	0.550
6000	0.8334	3	6.391	$2.427*10^{-1}$	0.600

TABLE B.2

Normalized Delay and Probability of Delay

for 250 Byte Data Packet

(Mean Transmission Time b_1 = 0.2304 msec)

λ (packets/sec)	D (msec)	K_{max}	D/b_1	Prob.	$\frac{\lambda * L_p}{v}$
120	0.2407	852	1.045	$< 10^{-39}$	0.624
200	0.2412	502	1.047	"	0.040
500	0.2435	186	1.057	"	0.100
1000	0.2485	81	1.079	"	0.200
1500	0.2557	45	1.110	$1.855*10^{-10}$	0.300
2000	0.2668	28	1.158	$1.325*10^{-6}$	0.400
2250		22		$2.349*10^{-5}$	0.450
2500	0.2864	17	1.243	$2.601*10^{-4}$	0.500
3000	0.3296	10	1.430	$7.826*10^{-3}$	0.600
3500	0.5034	5	2.185	$9.149*10^{-2}$	0.700
3750	1.387	3	6.023	$2.467*10^{-1}$	0.750

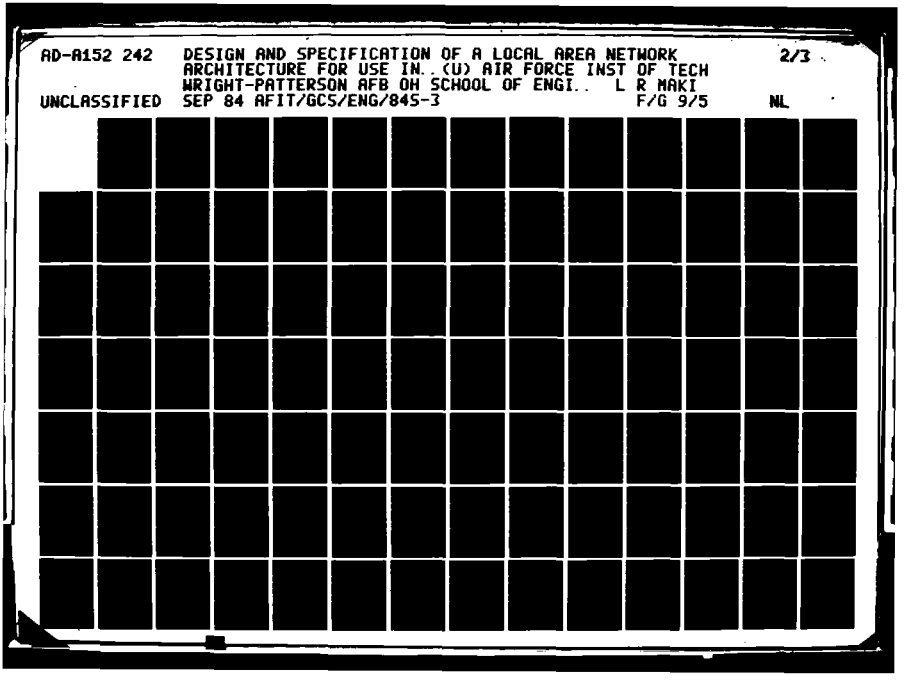
TABLE B.3

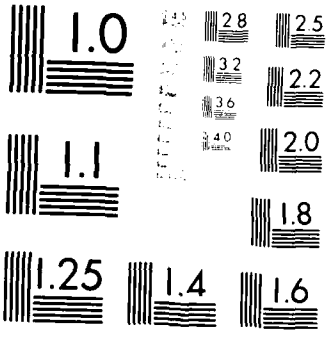
Normalized Delay and Probability of Delay

for 500 Byte Data Packet

(Mean Transmission Time b_1 = 0.4304 msec)

λ (packet/sec)	D (msec)	K_{max}	D/b_1	Prob.	$\frac{\lambda * L_p}{v}$
120	0.4414	831	1.025	$< 10^{-39}$	0.048
200	0.4424	481	1.028	"	0.080
250	0.4432	375	1.030	"	0.100
500	0.4474	165	1.039	"	0.200
750	0.4531	95	1.053	"	0.300
1000	0.4614	59	1.072	"	0.400
1250	0.4741	38	1.102	$1.512*10^{-8}$	0.500
1500	0.4965	24	1.153	$1.129*10^{-5}$	0.600
1625		19		$1.215*10^{-4}$	0.650
1750	0.5455	14	1.267	$1.300*10^{-3}$	0.700
2000	0.7393	7	1.718	$3.704*10^{-2}$	0.800
2125	1.657	4	3.849	$1.562*10^{-1}$	0.850
2150	3.265	3	7.585	$2.502*10^{-1}$	0.860


AD-A152 242 DESIGN AND SPECIFICATION OF A LOCAL AREA NETWORK ARCHITECTURE FOR USE IN..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. L R MAKI
2/3
UNCLASSIFIED SEP 84 AFIT/GCS/ENG/84S-3 F/G 9/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Appendix C: The Network Interface Board (NIB)

The network Interface board (NIB), designed by Lt. Richard Benken of the FCDL (Benken, 1984), acts as an interface between a host computer and an Ethernet LAN. It is based on two chips manufactured by Intel (the 82586 Local Communications Controller and the 82501 Ethernet Serial Interface) and a level switching driver (the 3Com transceiver). A block diagram of the prototype NIB is shown in Figure C.1. A more complete description of the functions provided by the 82586 and the 82501 can be found in Intel technical publications (Intel Corporation, 1982; and Intel Corporation, 1983). Basically, the two-chip set embody the CSMA/CD communications protocol of the Ethernet specification 1.0 (Digital Equipment Corporation and others, 1980), along with some diagnostics features. The remainder of this chapter provides a brief overview of the function of the NIB.

Basic Host-NIB Communications

The host computer communicates with the 82586 controller chip on the NIB not directly, but through a buffer memory located on the NIB. All data going into, and coming out from, the network passes through this memory. Furthermore, all commands to the 82586 from the host reside in this memory as well. On output, the host computer supplies a starting address and pulses a control line causing the NIB to step through the buffer memory addresses and fill the buffer memory with commands and data. The 82586 controller chip is then signalled to act

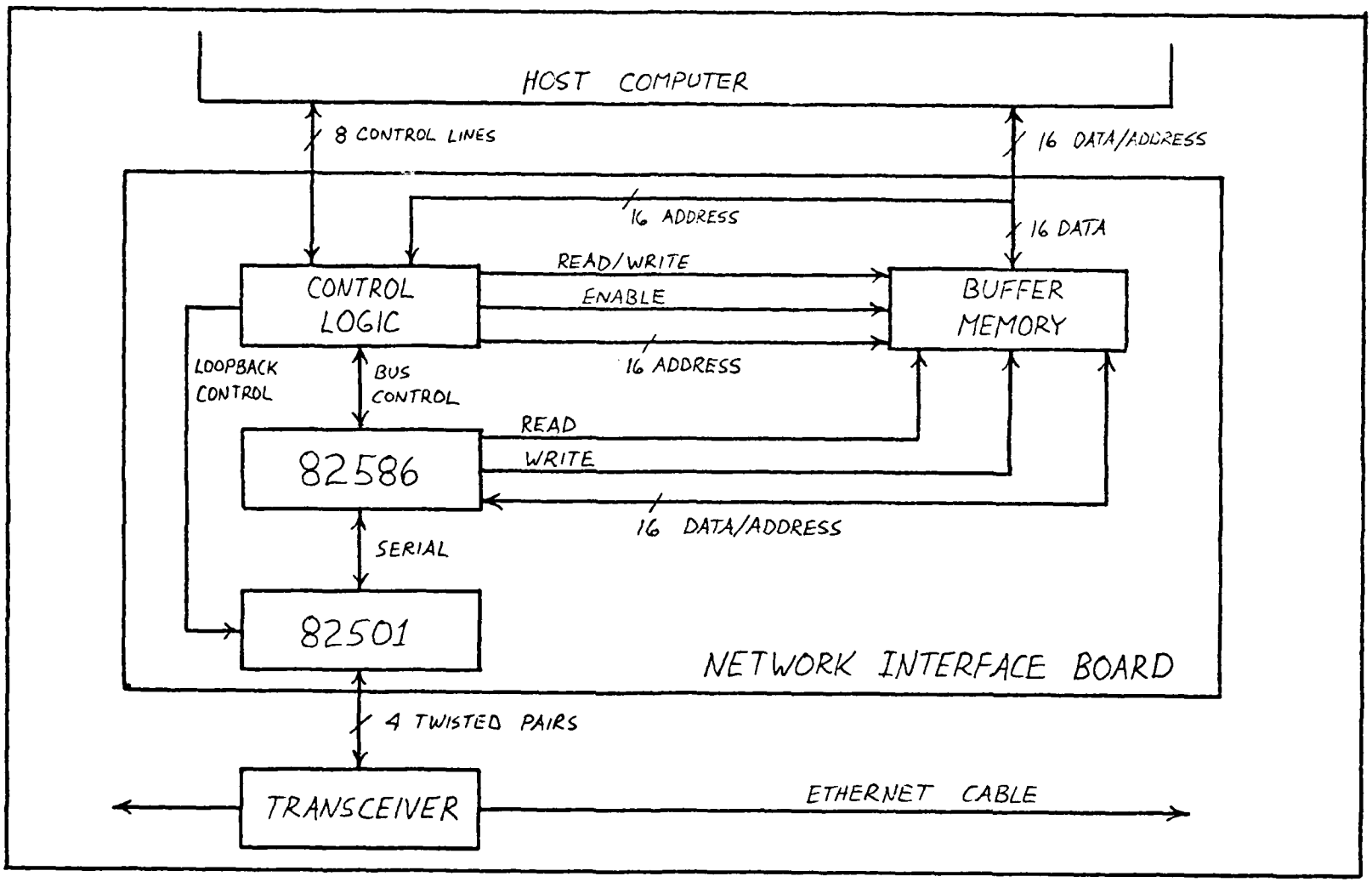


Figure C.1 Prototype NIB Block Diagram

upon these commands and data. On input, the 82586 fills the buffer memory (initialize by the host computer) with received data packets, and then signals the host computer of the reception. The host computer then accesses the buffer memory and pulses the data into its own memory.

Bus arbitration between the host computer and the 82586 is handled by giving the 82586 access to the bus at the end of any host computer memory cycle, whether or not the host computer has completed all transfers. If the 82586 does steal the bus, it signals the host computer through a control line. The host computer can then request the bus once again through another control line, and will be given control as soon as the 82586 finishes with its transfers.

NIB to Host Interface

The goal in the design of the NIB was to keep it generic enough to allowing interfacing to any device having a parallel port. The interface between the host and the NIB consists of 16 bidirectional data/address lines, and 8 control lines. The function of each control line is briefly described below.

RESET (input to NIB) - resets the 82586 and all necessary NIB internal logic to bring the NIB to a predetermined state.

INT (output from NIB) - signals the host processor of an error, receipt of data, or transmission of data.

D/$\overline{A}$ (input to NIB) - specifies whether the 16 bits on the bidirectional lines constitute data or an address.

R/$\overline{W}$ (input to NIB) - specifies whether to read or write. Together with the pevious input, D/A, four functions are specified:

- write data
- read data
- write address
- channel attention (a host to 82586 prompt)

$\overline{\text{DSSTR}}$ (input to NIB) - the data synchronization strobe has a dual purpose depending on whether the NIB is being read from or written to. In the read mode, it signals the NIB to output the data at the next sequential address. In the write mode it signals the NIB to strobe in the data on the bus into the present memory address location and then increment the address to the next sequential location.

$\overline{\text{BUSR}}$ (input to NIB) - signals the NIB that the host computer requests control of the bus.

$\overline{\text{BUSAV}}$ (output from NIB) - informs the host computer of the current status of bus availability; either the 82586 has control or the host has control. Since the 82586 has priority control, and can make ownership of the bus in the middle of a host transfer, the line is designed to immediately inform the host of such a change.

$\overline{\text{DACK}}$ (output from NIB) - the data acknowledge line has a dual purpose depending on whether the NIB is being read from or written to. In the read mode, it asserts a pulse while the NIB holds the data valid. In the write mode, it's pulse acknowledges receipt of the data or address.

NIB to Network Cable Interface

Using the Intel 82501 Ethernet Serial interface chip on the NIB, any Ethernet-compatible transceiver cable hardware can be used to interface the NIB to an Ethernet transceiver, and thus, the cable. The network hardware specified in Chapter II will be used for the FCDL network.

Appendix D: NIB Operations Control SADT Actigrams

This appendix presents the software design for the control of the NIB by an intelligent controller. The format adopted to present the design is the Structured Analysis and Design Technique (SADT) developed by SofTech (Softech, 1976). Included in the appendix are: a node index in Table D.1; a complete set of actigrams in index order, each with a page of explanatory text; and a data dictionary, Table D.2, defining the terms used in the actigrams.

It is assumed the reader is either familiar with the operation of the Intel 8258& Local Communications Controller, or has access to the technical manual describing its operation (Intel, 1983).

TABLE D.1

Node Index

TABLE D.1

Node Index (Continued)

TABLE D.1

Node Index (Continued)

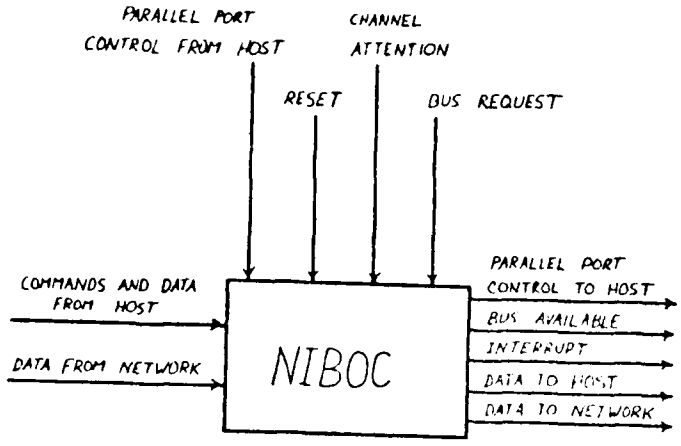
TABLE D.1

Node Index (Concluded)

A-0 NIBOC - Network Interface Board Operations Control

Abstract: This diagram is the environment node.

At this level, NIBOC is seen as a complete system. Inputs include commands and data from the host computer, and data received from the network. Control inputs consist of: parallel port controls from the host other than requesting the NIB bus; a reset input for resetting the NIB; a prompt from the host called channel attention; and a bus request input. Outputs include: parallel port controls to the host; an interrupt signal sent to the host; a bus available signal sent to the host; data to the host; and data to the network.



AØ NIBOC - NETWORK INTERFACE BOARD OPERATIONS CONTROL

A0 NIBOC - Network Interface Board Operations Control

Abstract: This diagram is the entire system decomposed along the five major functions of the system: Initialization, Configuration, System Diagnostics, Data Exchange, and Error Diagnostics. The execution of each activity is sequential, and activity Receive/Transmit Data is the primary activity during real-time operation. Note that control passes between the last two activities as errors occur.

Implementation details are shown to be functionally separated into five distinct processes. Initialization accepts commands and data from the host computer via parallel port control and bus request inputs, receives prompts reset and channel attention from the host, and performs the activity Initialize NIB. This activity generates parallel port control outputs to the host, tells the host the bus is available, and outputs an interrupt to the host signalling activity completion.

Configuration performs the activity Configure NIB, which takes commands and data from the host computer via parallel port and control and bus request inputs, receives a channel attention prompt from the host, and performs the activity Configure NIB. This activity generates parallel port control outputs to the host, tells the host the bus is available, and outputs an interrupt to the host signalling activity completion.

System Diagnostics takes commands and data from the host computer and data from the network, along with parallel port, bus request, and channel attention prompts from the host, and performs the activity Diagnose NIB/Network System. This activity generates parallel port control outputs to the host, tells the host of bus availability, outputs data to the host, and outputs an interrupt to the host signalling activity completion.

Network Data Exchange takes commands and data from the host and data from the network, along with parallel port, bus request, and channel attention prompts from the host, and performs the activity Receive/Transmit Data. This activity generates data to the host, data to the network, parallel port control outputs to the host, tells the host of bus availability, and outputs interrupts to the host signalling activity completion. The interrupt output is also used to control transfer of activity between Network Data Exchange and Error Diagnostics.

Error Diagnostics takes commands and data from the host computer via parallel port, bus request, channel attention, and interrupt prompts from the host, and performs the activity Diagnose NIB Errors. This activity generates data to the host, tells the host of bus availability, outputs parallel port controls to the host, and outputs interrupts to the host signalling activity completion. The interrupt output is also used to control transfer of activity between Network Data Exchange and Error Diagnostics.

A23 Select Configuration

Abstract: This diagram decomposes the activity Select Configuration into its major functions.

Select Configuration requires the host to create in NIB memory a Command Block List (CBL) containing the Configure command. The 82586 System Control Block (SCB) is then prepared so that , when channel attention is applied, the 82586 knows to execute the Configure command. Execution of A231, A232, A233, A234, and A235 is sequential and is implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Select Configuration, the 82586 does take control as it executes the command. However, the host is not asking for the bus at this time, so there is no contention.

A231 (Identical to activity A211)

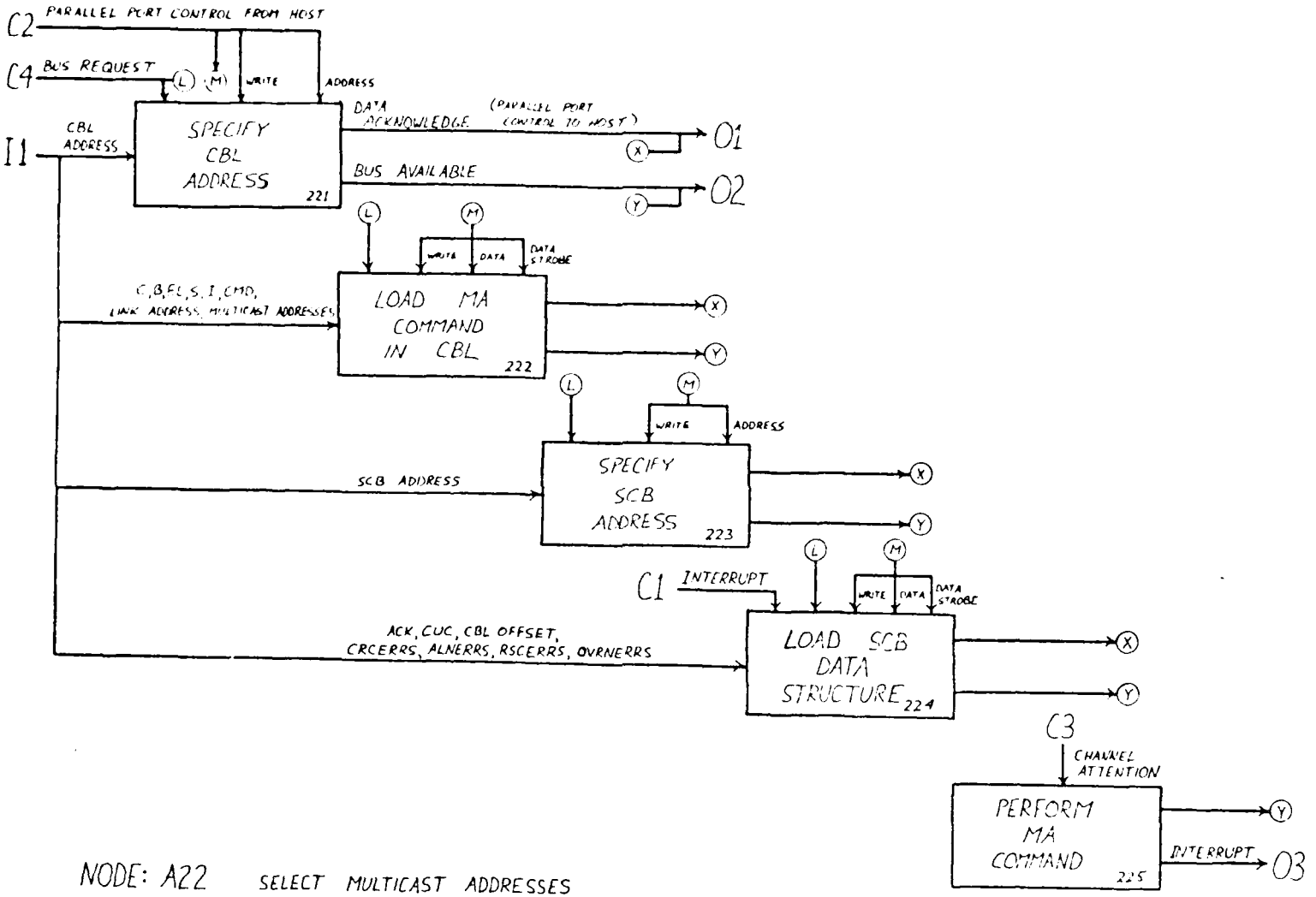
A232 Load Configure Command In CBL loads the Configure parameters shown as inputs into the CBL located by activity A231. After requesting the bus, the host loads the Configure command parameters (according to the Configure command data structure) into the CBL via the parallel port controls write, data, and data strobe. The 82586 responds with data acknowledgements.

A233 (Identical to activity A213)

A234 Load SCB Data Structure loads the SCB parameters shown into the SCB data structure at the address specified in A233. After requesting the bus, the host loads this data into the SCB data structure via the parallel port controls write, data, and data strobe. Note that the interrupt from either activity A13, activity A215, or activity A225 (whichever one preceeded A23) must be acknowledged in the input ACK. The 82586 responds to this activity with data acknowledgements.

A235 Perform Configure Command is the execution of the Configure command by the 82586. It is begun by the host applying channel attention to the NIB. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the Configure CBL).

Again, the execution of these activities must be in the order A231, A232, A233, A234, A235.



NODE: A22 SELECT MULTICAST ADDRESSES

A22 Select Multicast Addresses

Abstract: This diagram decomposes the activity Select Multicast Addresses into its major functions.

Select Multicast Addresses requires the host to create in NIB memory a Command Block List (CBL) containing the Multicast Address (MA) command. The 82586 System Control Block (SCB) is then prepared so that, when channel attention is applied, the 82586 knows to execute the MA command. Execution of A221, A222, A223, A224, and A225 is sequential and is implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Select Multicast Addresses, the 82586 does take control as it executes the command. However, the host is not asking for the bus at this time so there is no contention.

A221 (Identical to activity A211)

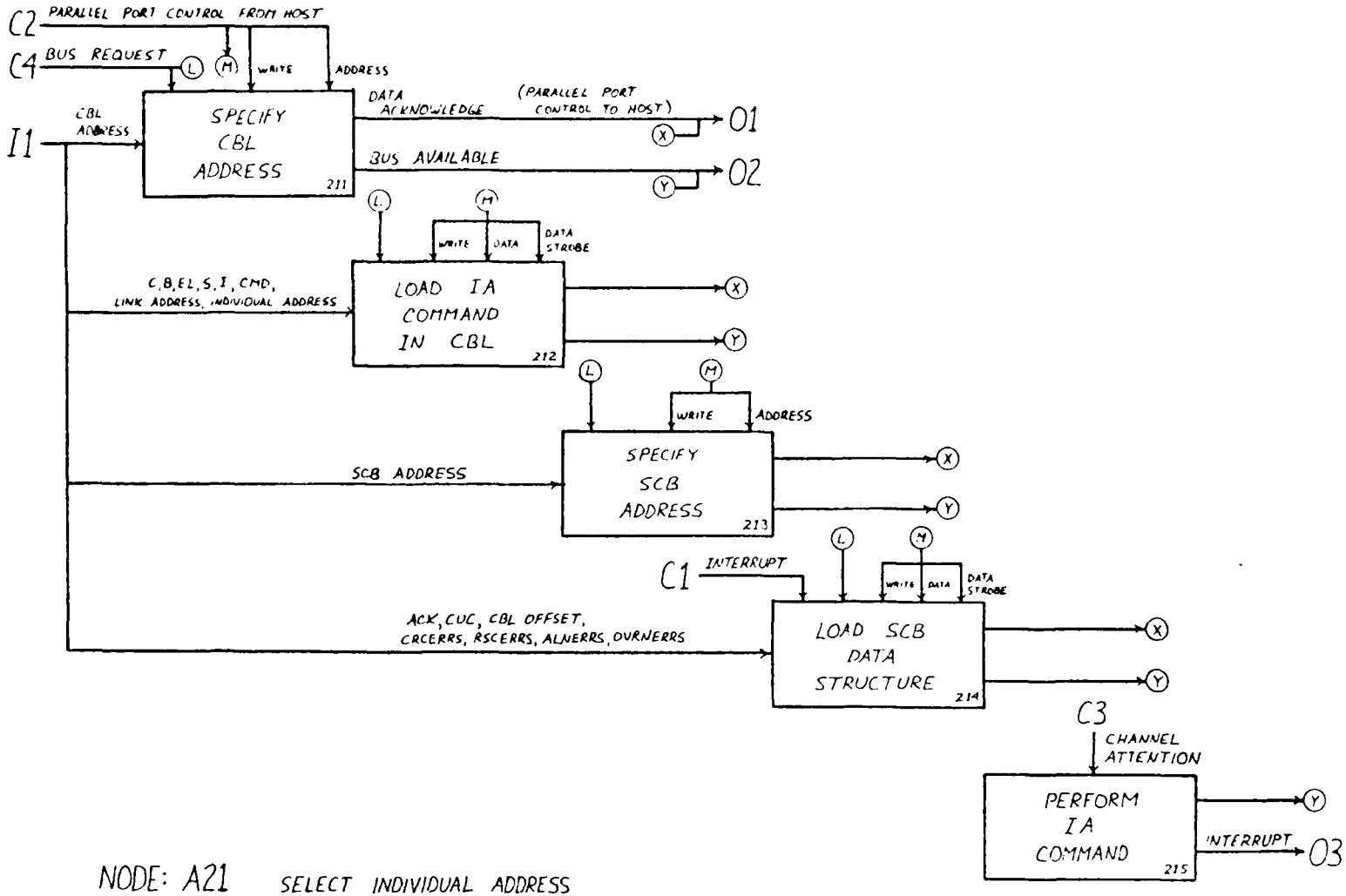
A222 Load MA Command In CBL loads the MA parameters shown as inputs into the CBL located by activity A221. After requesting the bus, the host loads the MA command parameters (according to the MA command data structure) into the CBL via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.

A223 (Identical to activity A213)

A224 Load SCB Data Structure loads the SCB Parameters shown into the SCB data structure at the address specified in A223. After requesting the bus, the host loads this data into the SCB data structure via the parallel port controls write, data, and data strobe. Note that the interrupt from either activity A13 or activity A215 (whichever one preceeded A22) must be acknowledged in the input ACK. The 82586 responds to this activity with data acknowledgements.

A225 Perform MA Command is the execution of the MA command by the 82586. It is begun by the host applying channel attention to the NIB. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the MA CBL).

Again, the execution of these activities must be in the order A221, A222, A223, A224, A225.



NODE: A21 SELECT INDIVIDUAL ADDRESS

A21 Select Individual Address

Abstract: This diagram decomposes the activity Select Individual Address into its major functions.

Select Individual Address requires the host to create in NIB memory a Command Block List (CBL) containing the Individual Address (IA) command. The 82586 System Control Block (SCB) is then prepared so that, when channel attention is applied, the 82586 knows to execute the IA command. Execution of A211, A212, A213, A214, and A215 is sequential and is implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Select Individual Address, the 82586 does take control as it executes the command. However, the host is not asking for the bus at this time so there is no contention.

A211 Specify CBL Address is performed by the host prior to loading data into the address specified. After requesting the bus, the host applies the CBL address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

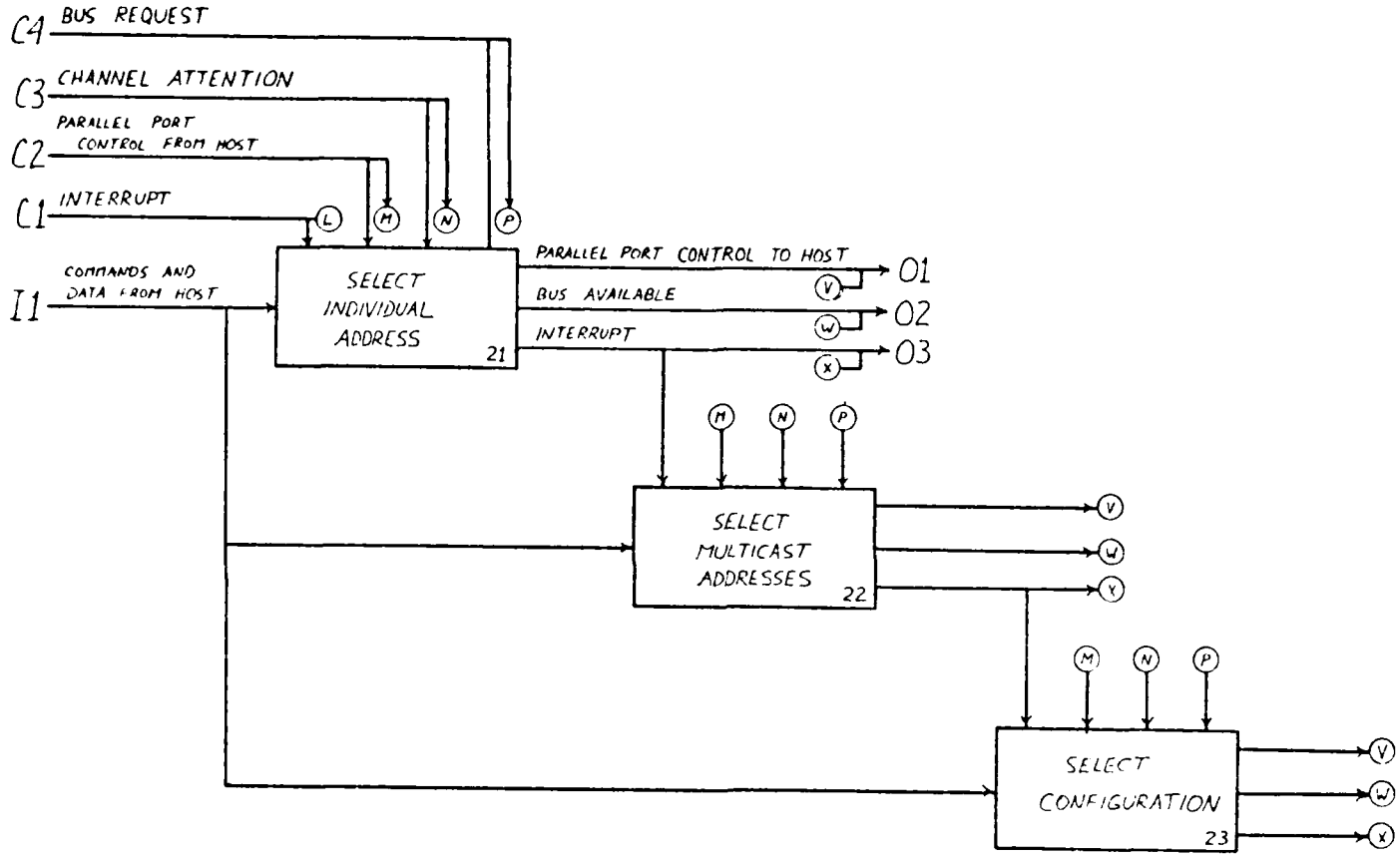
A212 Load IA Command In CBL loads the IA parameters shown as inputs into the CBL located by activity A211. After requesting the bus, the host loads the IA command parameters (according to the IA command data structure) into the CBL via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.

A213 Specify SCB Address is performed by the host prior to loading data into the SCB. After requesting the bus, the host applies the SCB address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

A214 Load SCB Data Structure loads the SCB parameters shown into the SCB data structure at the address specified in A213. After requesting the bus, the host loads this data into the SCB data structure via the parallel port controls write, data, and data strobe. Note that the interrupt from activity A13 must be acknowledged in the input ACK. The 82586 responds to this activity with acknowledgements.

A215 Perform IA Command is the execution of the IA command by the 82586. It is begun by the host applying channel attention to the NIB. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the IA CBL).

Again, the execution of these activities must be in the order A211, A212, A213, A214, A215.



NODE: A2 CONFIGURE NIB

A2 Configure NIB

Abstract: This diagram decomposes the activity of Configuration into its major functions.

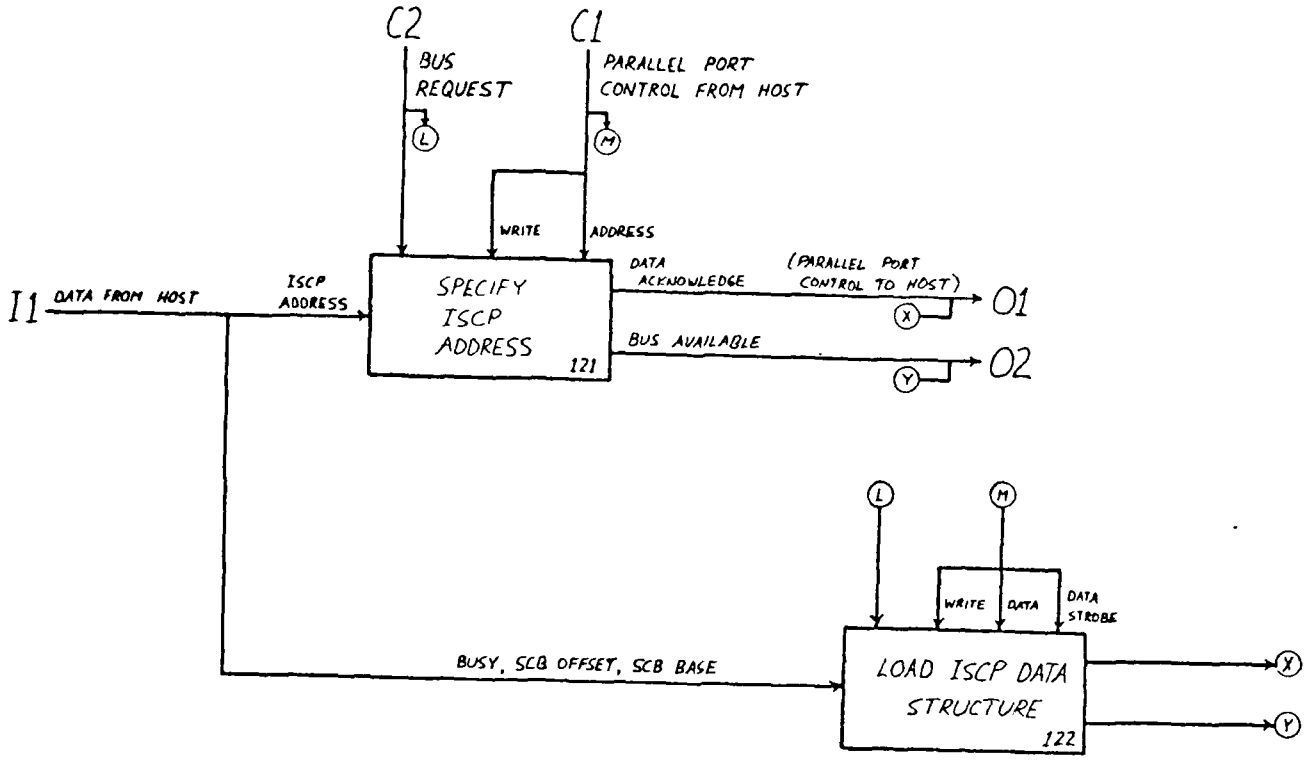
Configuration is composed of activities which give the NIB its identity and tailor its operation to variations of the Ethernet protocol. Although the order of execution of the activities A21, A22, and A23 is implied by the interrupt output of one activity feeding in as a control input on the next activity, the execution can be in any order. However, activity Select Individual Address must be accomplished to ensure the NIB of an identity. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Configure NIB, the 82586 does take control of the bus during the execution of the commands. However, the host is not asking for the bus at these times, so there is no contention. The interrupt output from each activity is typically the result of the host setting the interrupt bit in a command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host before the next activity can be completed. Therefore, interrupt is shown as a control on each activity.

A21 Select Individual Address assigns a unique (up to) 6 byte address to the NIB. It uses this address to identify data packets intended for it, and to indicate the source of data packets sent by it. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with parallel port controls while accepting commands and data, and signals an interrupt when finished.

A22 Select Multicast Addresses, in a manner similar to Select Individual Addresses, assigns one or more addresses by which the NIB can be identified. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with parallel port controls while accepting commands and data, and signals an interrupt when finished.

A23 Select Configuration handles the setting of parameters which vary the protocol from the default Ethernet standard. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with parallel port controls while accepting commands and data, and signals an interrupt when finished.

Again, the sequence of these activities is not important, but the execution of Select Individual Address is the minimum requirement.



NODE: A12 SET UP 82586 ISCP

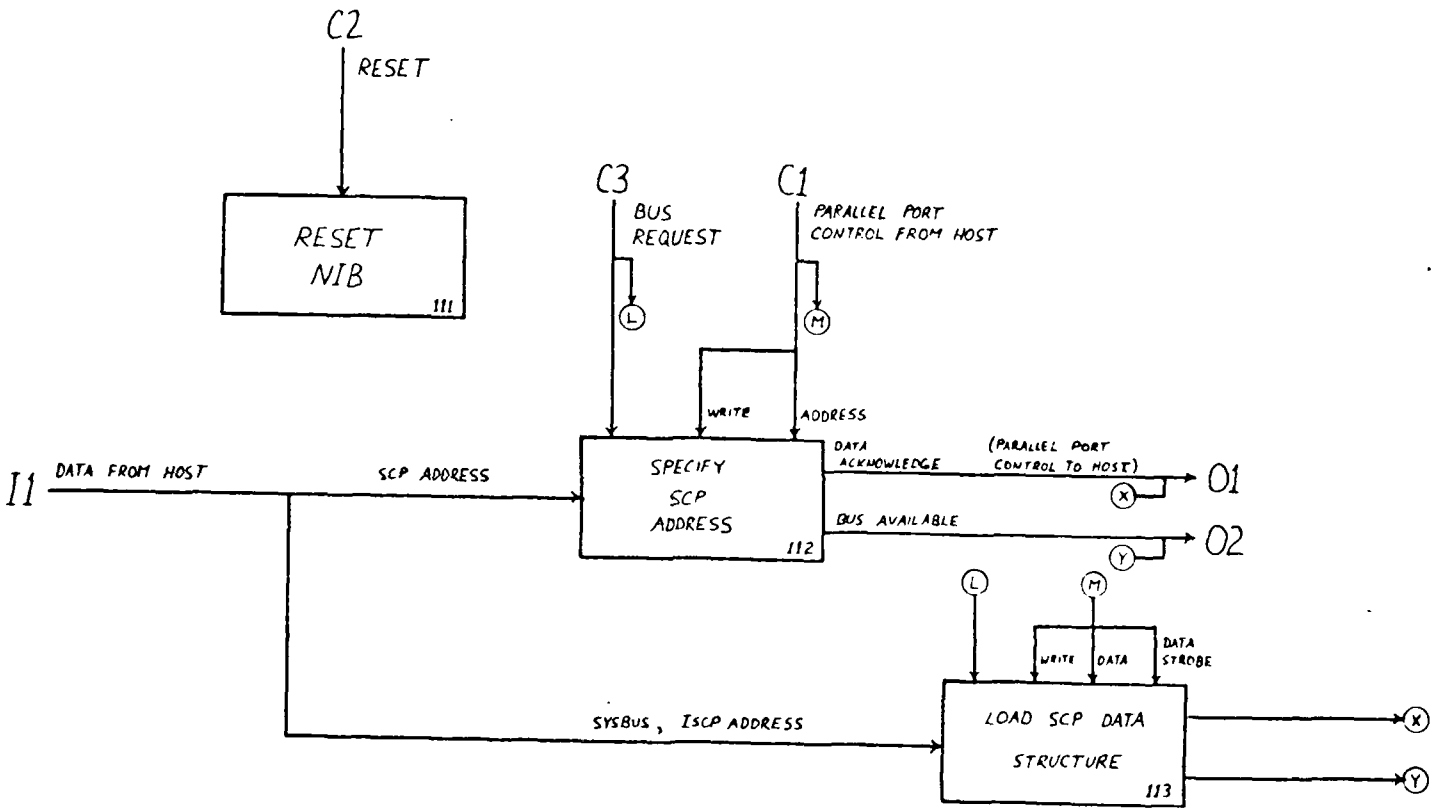
A12 Set Up 82586 ISCP

Abstract: This diagram decomposes the activity Set Up 82586 ISCP into its major functions.

Set Up 82586 ISCP requires the specification of the ISCP starting address, and the loading of the data into the ISCP data structure. Execution of A121 and A122 is sequential and is implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Set Up 82586 ISCP, the 82586 does not ever take control as it is still being prepared.

A121 Specify ISCP Address is performed by the host to notify the 82586 of the location of the ISCP data structure. After requesting the bus, the host applies the ISCP address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

A122 Load ISCP Data Structure loads the values of Busy, SCB Offset, and SCB Base into the ISCP. After requesting the bus, the host loads this data into the ISCP data structure via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.



NODE: A11 SET UP 82586 SCP

A11 Set Up 82586 SCP

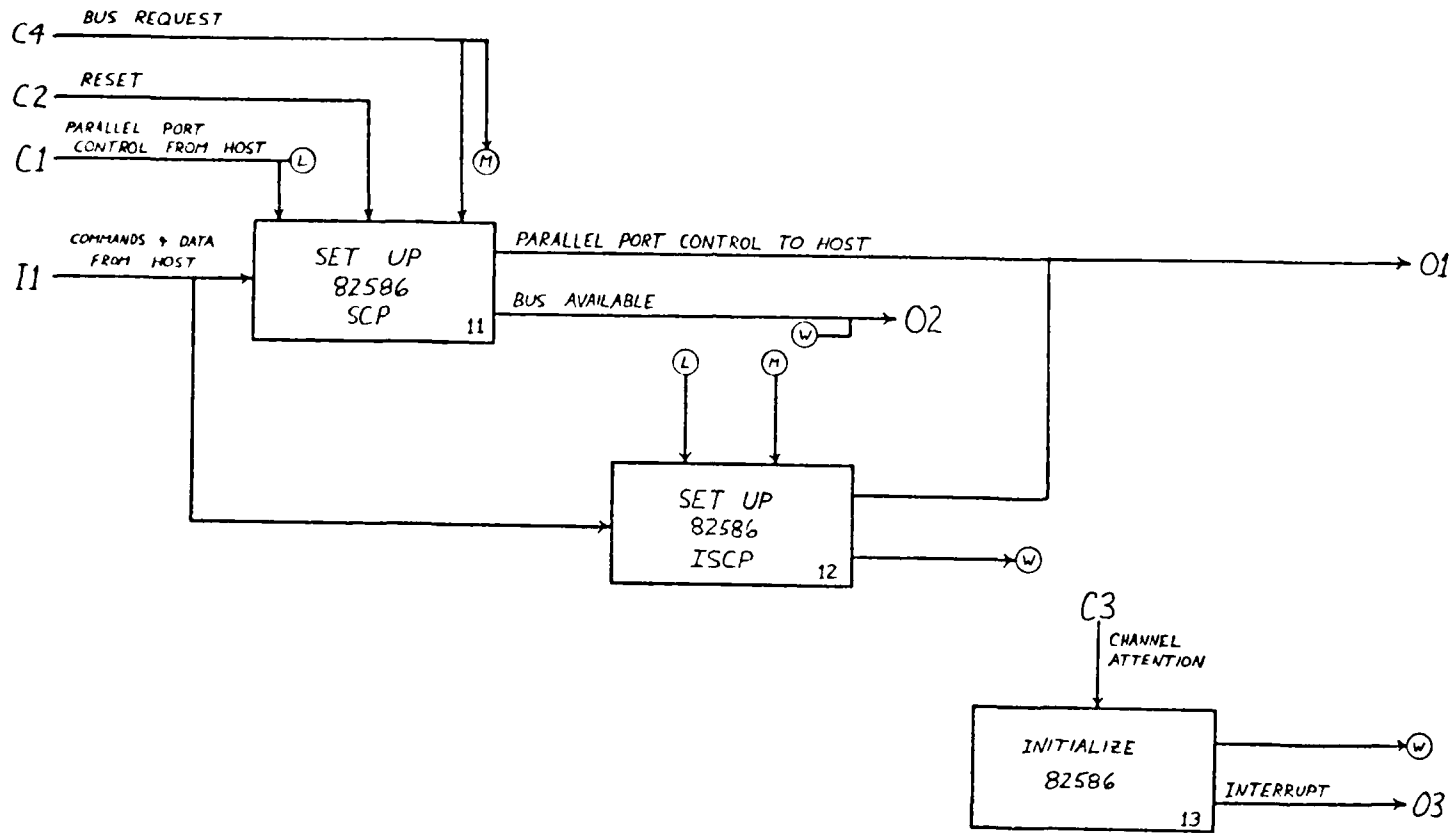
Abstract: This diagram decomposes the activity Set Up 82586 SCP into its major functions.

Set Up 82586 SCP requires the resetting of the 82586, specification of the SCP starting address, and the loading of the data into the SCP data structure. Execution of A111, A112, and A113 is sequential and is implied by the their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Set Up 82586 SCP, the 82586 does not ever take control as it is still being prepared.

A111 Reset NIB is a function handled by the 82586 and initiated by the host applying reset to the NIB. It prepares the 82586 for the remaining steps in initialization.

A112 Specify SCP Address is performed by the host to notify the 82586 of the location of the SCP data structure. After requesting the bus, the host applies the SCP address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

A113 Load SCP Data Structure loads the values of Sysbus and the location of the ISCP into the SCP. After requesting the bus, the host loads this data into the SCP data structure via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.



NODE: A1 INITIALIZE NIB

A1 Initialize NIB

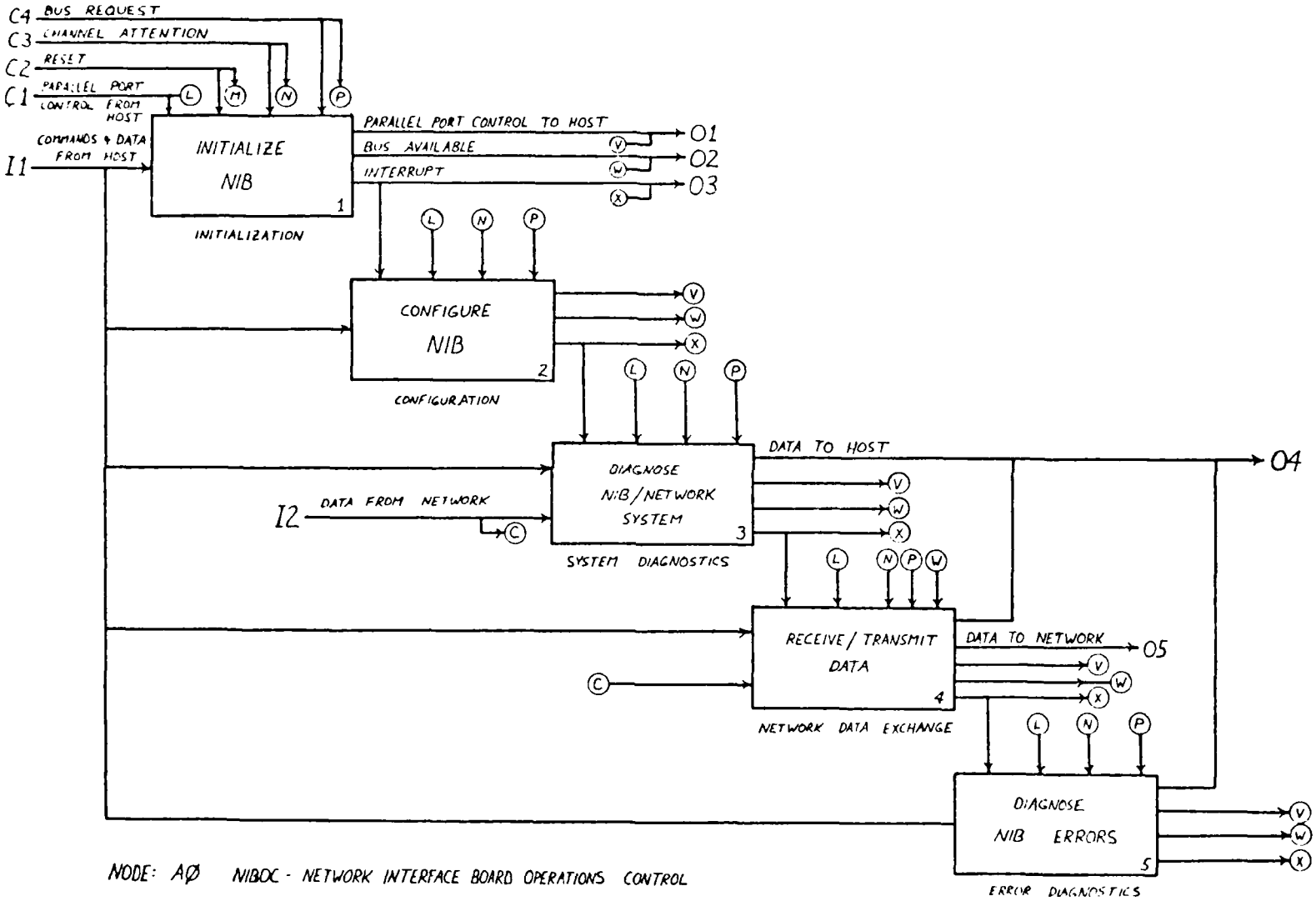
Abstract: This diagram decomposes the activity of Initialization into its major functions.

Initialization requires the preparation of two 82586 data structures, the System Configuration Pointer (SCP) and the Intermediate System Control Pointer (ISCP). Execution of A11, A12, and A13 is sequential and is implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Initialize NIB, the 82586 does take control as it does the initialization (activity A13). However, the host is not asking for the bus at this time so there is no contention.

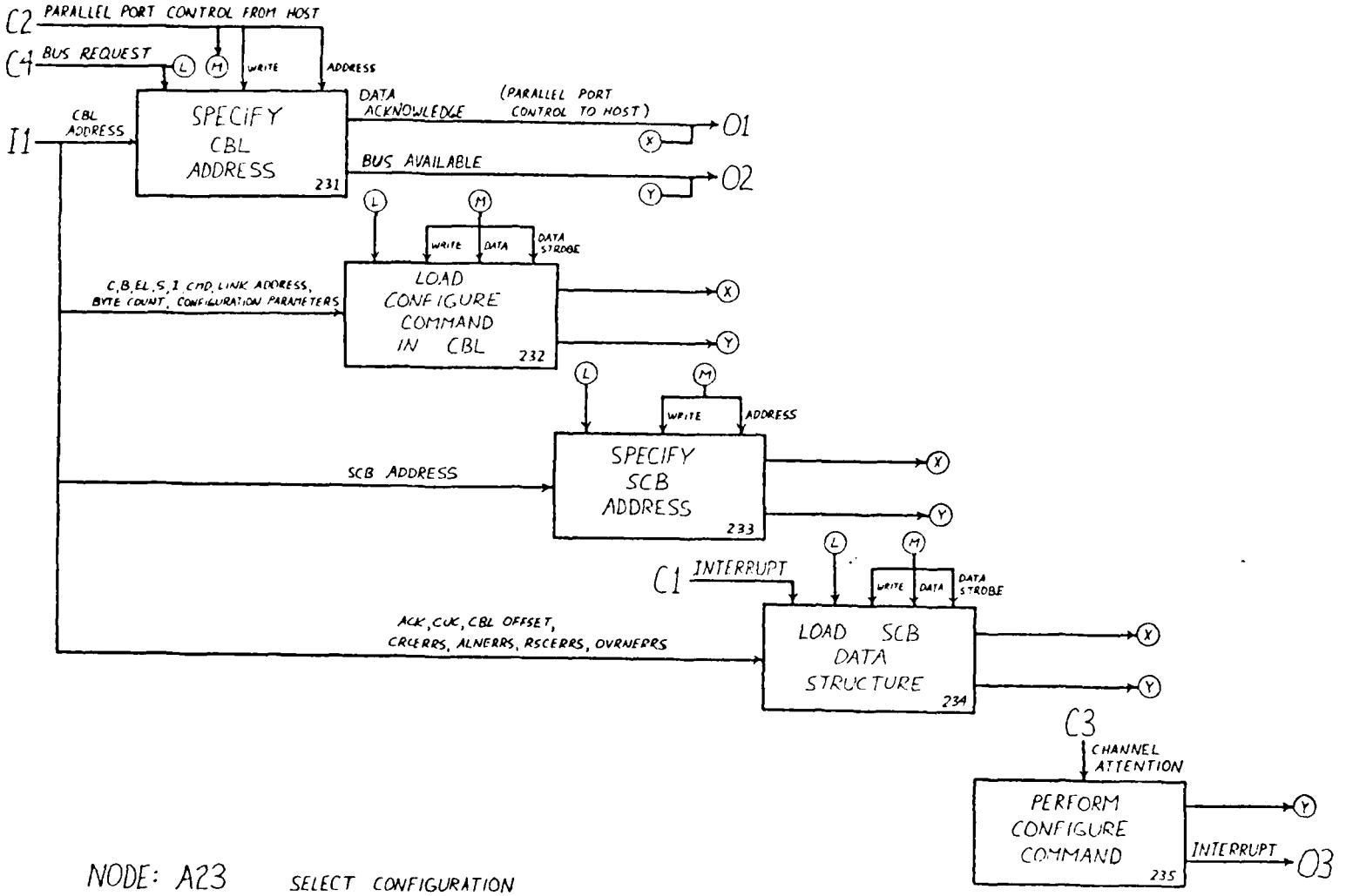
A11 Set Up 82586 SCP handles the preparation of the 82586 SCP by the host computer. After applying a reset, the NIB bus is requested by the host. Appropriate commands and data are issued to the 82586 SCP via parallel port controls to and from the host.

A12 Set Up 82586 ISCP handles the preparation of the 82586 ISCP by the host computer. The NIB bus is first requested by the host, and then appropriate commands and data are issued to the 82586 ISCP via parallel port controls to and from the host.

A13 Initialize 82586 is the actual execution of the 82586 initialization process internal to the 82586. Channel attention is applied by the host, and the 82586 indicates completion via the interrupt output.



NODE: AØ NIBOC - NETWORK INTERFACE BOARD OPERATIONS CONTROL



NODE: A23 SELECT CONFIGURATION

A3 Diagnose NIB/Network System

Abstract: This diagram decomposes the activity of System Diagnostics into its major functions.

System Diagnostics is composed of activities which test the operation of the 82586 and the integrity of the network. Although the order of execution of the activities A31, A32, A33, and A34 is implied by the interrupt output of one activity feeding in as a control input on the next activity, the execution can be in any order. However, the order as shown is recommended since additional hardware is tested at each step. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity A3, the 82586 does take control of the bus during the execution of the commands. However, the host is not actively asking for the bus at these times, so there is no contention. The interrupt output from each activity is typically the result of the host setting the interrupt bit in a command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host before the next activity can be completed. Therefore, interrupt is shown as a control on each activity.

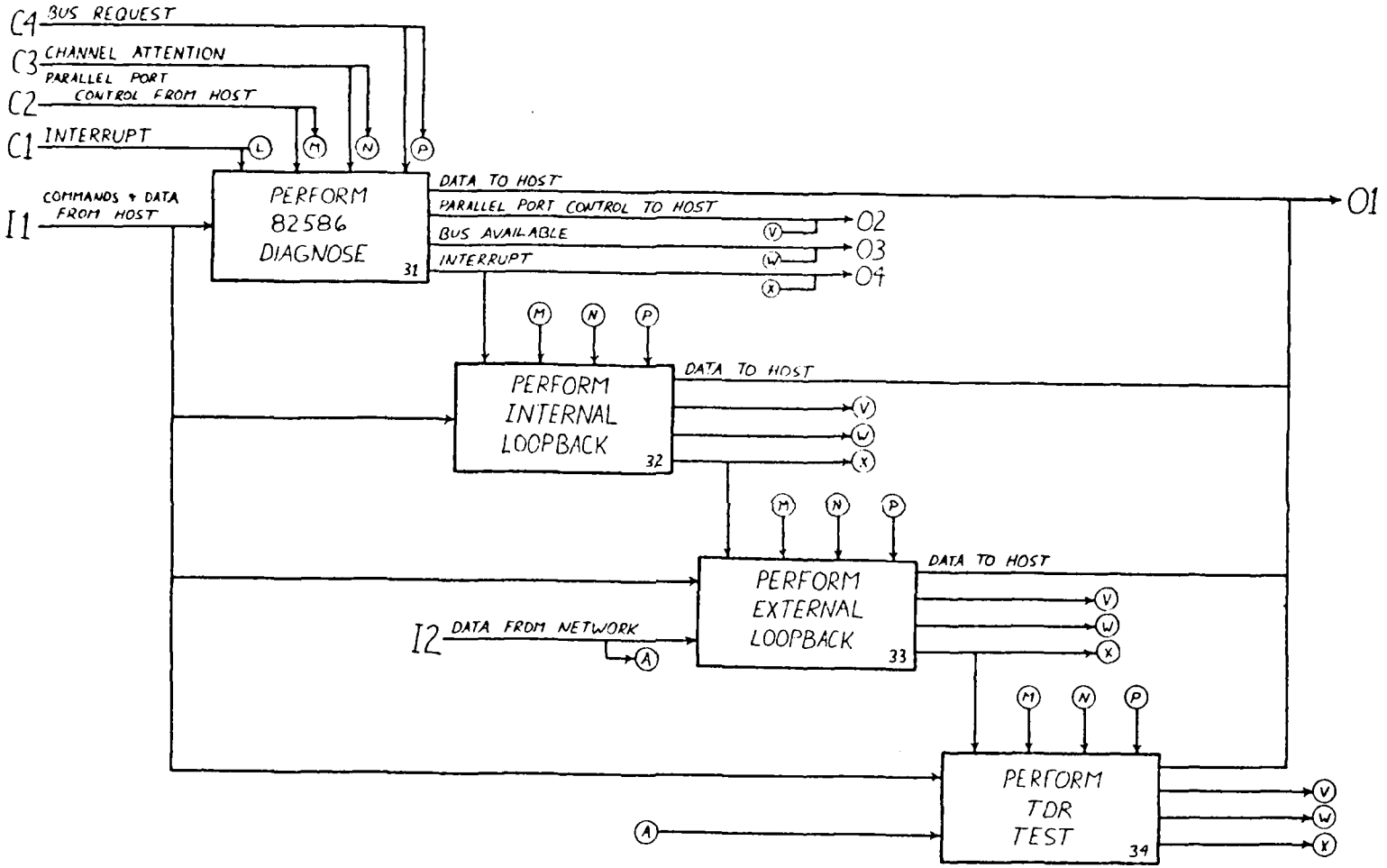
A31 Perform 82586 Diagnose executes a diagnostic test of the 82586 internal timer hardware. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with command execution. The results of the test are then collected by the host for analysis.

A32 Perform Internal Loopback executes a diagnostic test of the transmission and reception functions of the 82586 internally. The description of the inputs and outputs is the same as for A31. The results of the test are then collected by the host for analysis.

A33 Perform External Loopback executes a diagnostic test of the transmission and reception functions of the 82586 for a frame limited to 18 bytes. Unlike activity A32, however, activity A33 allows the checking of external hardware as well as the serial link to the transceiver. The description of the inputs and outputs is the same as for A31. During command execution, data from the network (the data sent by the NIB) is received by the NIB. The results of the test are then collected by the host for analysis.

A34 Perform TDR Test executes a time domain reflectometer test on the serial link. By performing the test, shorts or opens on the network serial link, and their location, can be identified. The description of the inputs and outputs is the same as for A31. During command execution, data from the network (the reflected signals, if any) is received by the NIB. The results of the test are then collected by the host for analysis.

Again, the order of execution of activities A31, A32, A33, and A34 is not important, but the order shown is recommended.



NODE: A3 DIAGNOSE NIB/NETWORK SYSTEM

A31 Perform 82586 Diagnose

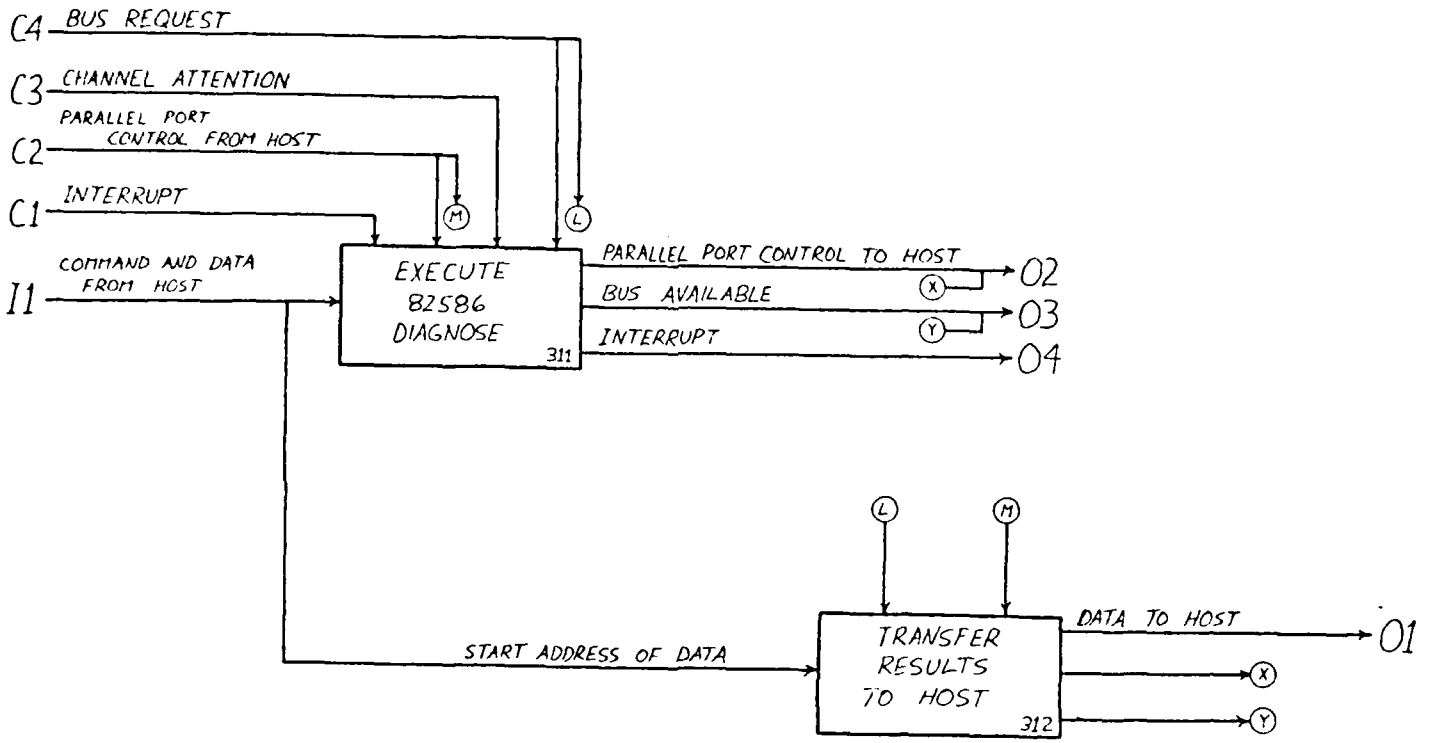
Abstract: This diagram decomposes the activity Perform 82586 Diagnose into its major functions.

Perform 82586 Diagnose is composed of activities which set up and execute the 82586 diagnose command and transfer the results to the host. The order of execution of activities A311 and A312 is sequential as implied by their positioning from upper left to lower right.

A311 Execute 82586 Diagnose performs the internal 82586 timer test. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The host must also acknowledge the interrupt (shown as a control input) that was generated by the NIB upon completion of a previous activity (such as A235). The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with command execution. The bus available output from this activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Execute 82586 Diagnose, the 82586 does take control of the bus during the execution of the commands. However, the host is not actively asking for the bus at these times, so there is no contention. The interrupt output from this activity is typically the result of the host setting the interrupt bit in the command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host before the next activity prompted by a channel attention can be completed.

A312 Transfer Results To Host handles the movement of data from the NIB memory to the host. After requesting the bus, the host uses parallel port controls to input the starting address of the data to be transferred and to step through the NIB memory. The NIB responds with parallel port controls to the host for synchronization, and with the data. The bus available output is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Transfer Results To Host, the 82586 does not ever take control, so there is no contention problem.

Again, the execution of these activities must be in the order A311, A312.



NODE: A31 PERFORM 82586 DIAGNOSE

A311 Execute 82586 Diagnose

Abstract: This diagram decomposes the activity Execute 82586 Diagnose into its major functions.

Execute 82586 Diagnose requires the host to create in NIB memory a Command Block List (CBL) containing the Diagnose command. The 82586 System Control Block (SCB) is then prepared so that, when channel attention is applied, the 82586 knows to execute the Diagnose command. Execution of A3111, A3112, A3113, A3114, and A3115 is sequential and is implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Execute 82586 Diagnose, the 82586 does take control as it executes the command. However, the host is not asking for the bus at this time, so there is no contention.

A3111 (Identical to activity A211)

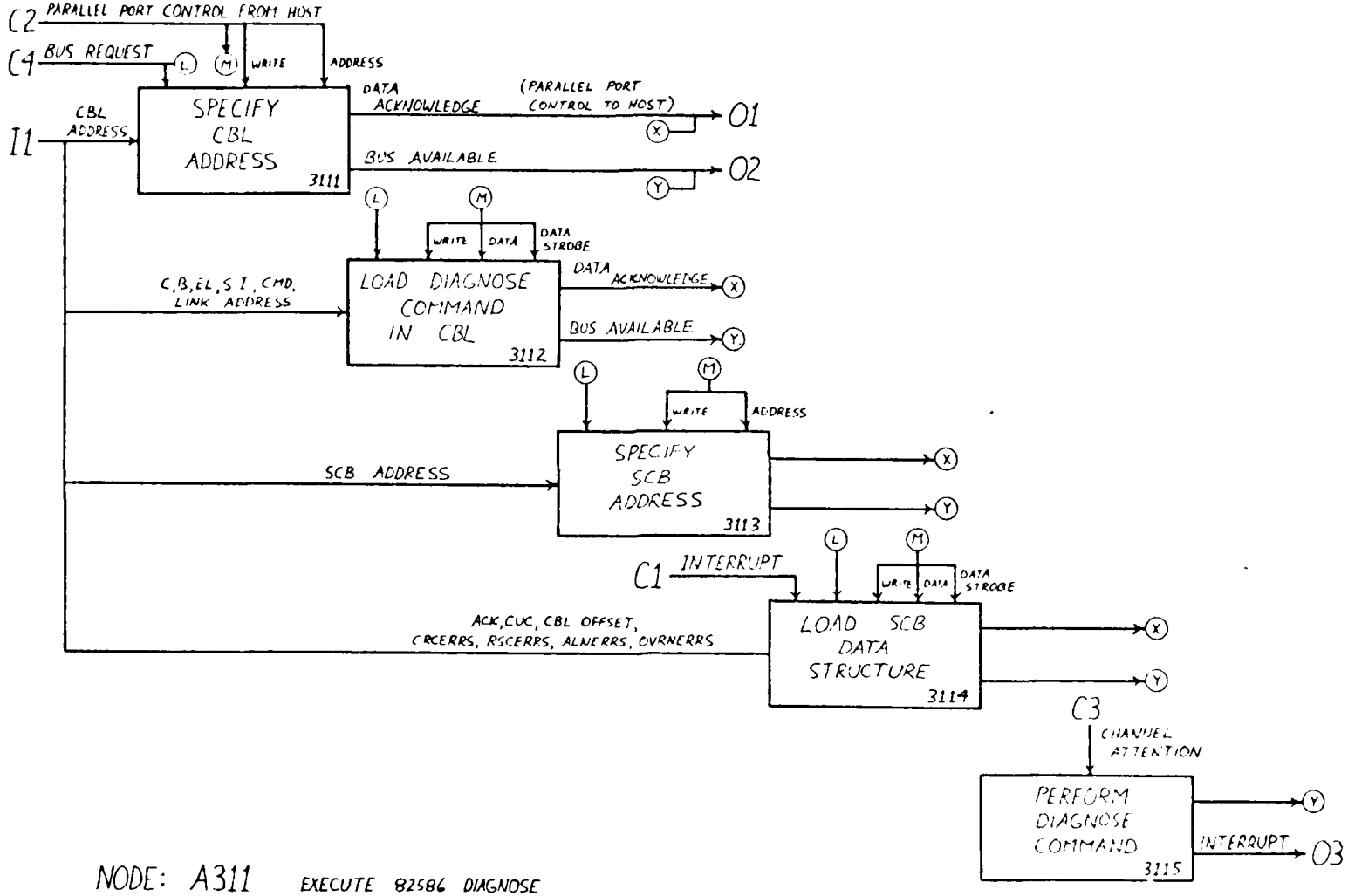
A3112 Load Diagnose Command In CBL loads the Diagnose parameters shown as inputs into the CBL located by activity A3111. After requesting the bus, the host loads the Diagnose command parameters (according to the Diagnose command data structure) into the CBL via the parallel port controls write, data, and data strobe. The 82586 responds with data acknowledgements.

A3113 (Identical to activity A213)

A3114 Load SCB Data Structure loads the SCB parameters shown into the SCB data structure at the address specified in A3113. After requesting the bus, the host loads this data into the SCB data structure via the parallel port controls write, data, and data strobe. Note that the interrupt generated from a previous activity must be acknowledged in the input ACK. The 82586 responds to this activity with data acknowledgements.

A3115 Perform Diagnose Command is the execution of the Diagnose command by the 82586. It is begun by the host applying channel attention to the NIB. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the Diagnose CBL).

Again, the execution of these activities must be in the order A3111, A3112, A3113, A3114, A3115.



NODE: A311 EXECUTE 82586 DIAGNOSE

A312 Transfer Results To Host

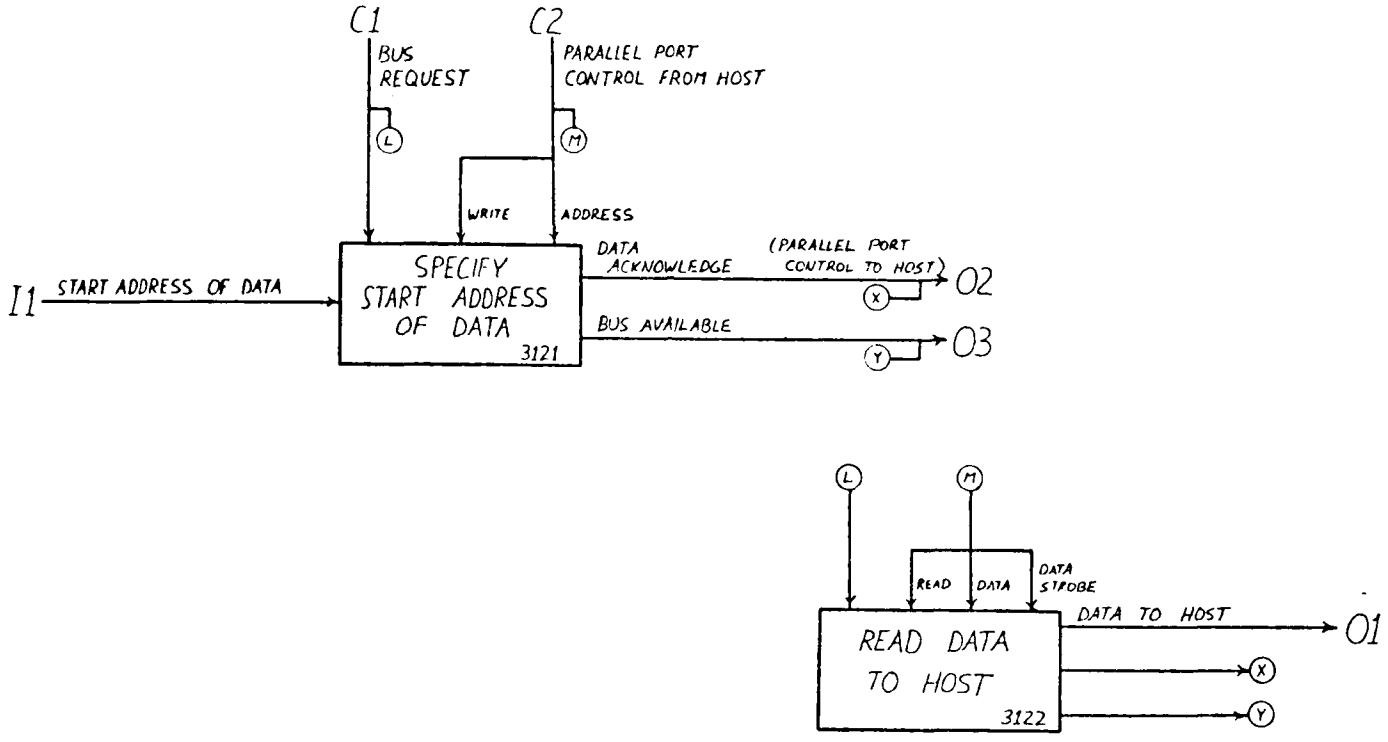
Abstract: This diagram decomposes the activity Transfer Results To Host into its major functions.

Transfer Results To Host is composed of activities which locate the data of interest and move it to the host. The order of execution of activities A3121 and A3122 is sequential as implied by their positioning from upper left to lower right. The bus available output is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Transfer Results To Host, the 82586 does not ever take control, so there is no contention problem.

A3121 Specify Start Address Of Data locates the starting address of the data, and is performed by the host prior to reading the data from the NIB memory. After requesting the bus, the host applies the data starting address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

A3122 Read Data To Host handles the actual function of transferring the data. After requesting the bus, the host uses the parallel port controls read, data, and data strobe to step through NIB memory and read the data. Data is transferred to the host, and the NIB responds with data acknowledgements.

Again, the execution of these activities must be in the order A3121, A3122.



NODE: A312 TRANSFER RESULTS TO HOST

A32 Perform Internal Loopback

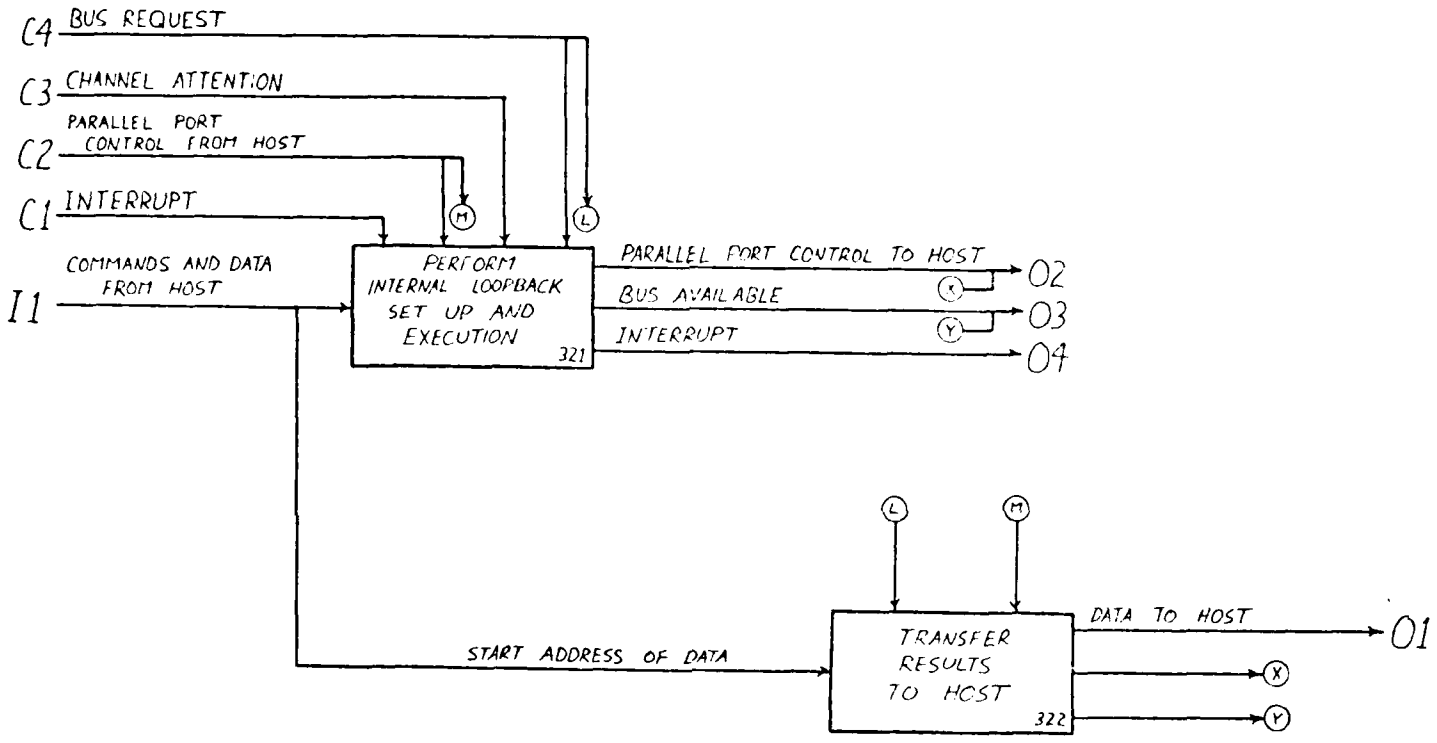
Abstract: This diagram decomposes the activity Perform Internal Loopback into its major functions.

Perform Internal Loopback is composed of activities which set up and execute the 82586 internal loopback test and transfer the results to the host. The order of execution of activities A321 and A322 is sequential as implied by their positioning from upper left to lower right.

A321 Perform Internal Loopback Set Up And Execution prepares the NIB to do the internal loopback test and then executes it. After requesting the bus, the host uses parallel port controls to input commands and data for preparing and executing this activity. The host must also acknowledge the interrupt (shown as a control input) that was generated by the NIB upon completion of a previous activity. The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with command execution. The bus available output from this activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Perform Internal Loopback Set Up And Execution, the 82586 does take control of the bus during the execution of the commands. However, the host is not actively asking for the bus at these times, so there is no contention. The interrupt output from this activity is typically the result of the host setting the interrupt bit in the command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host before the next activity prompted by a channel attention can be completed.

A322 (Identical to activity A312)

Again, the execution of these activities must be in the order A321, A322.



NODE: A32 PERFORM INTERNAL LOOPBACK

A321 Perform Internal Loopback Set Up And Execution

Abstract: This diagram decomposes the activity Perform Internal Loopback Set Up And Execution into its major functions.

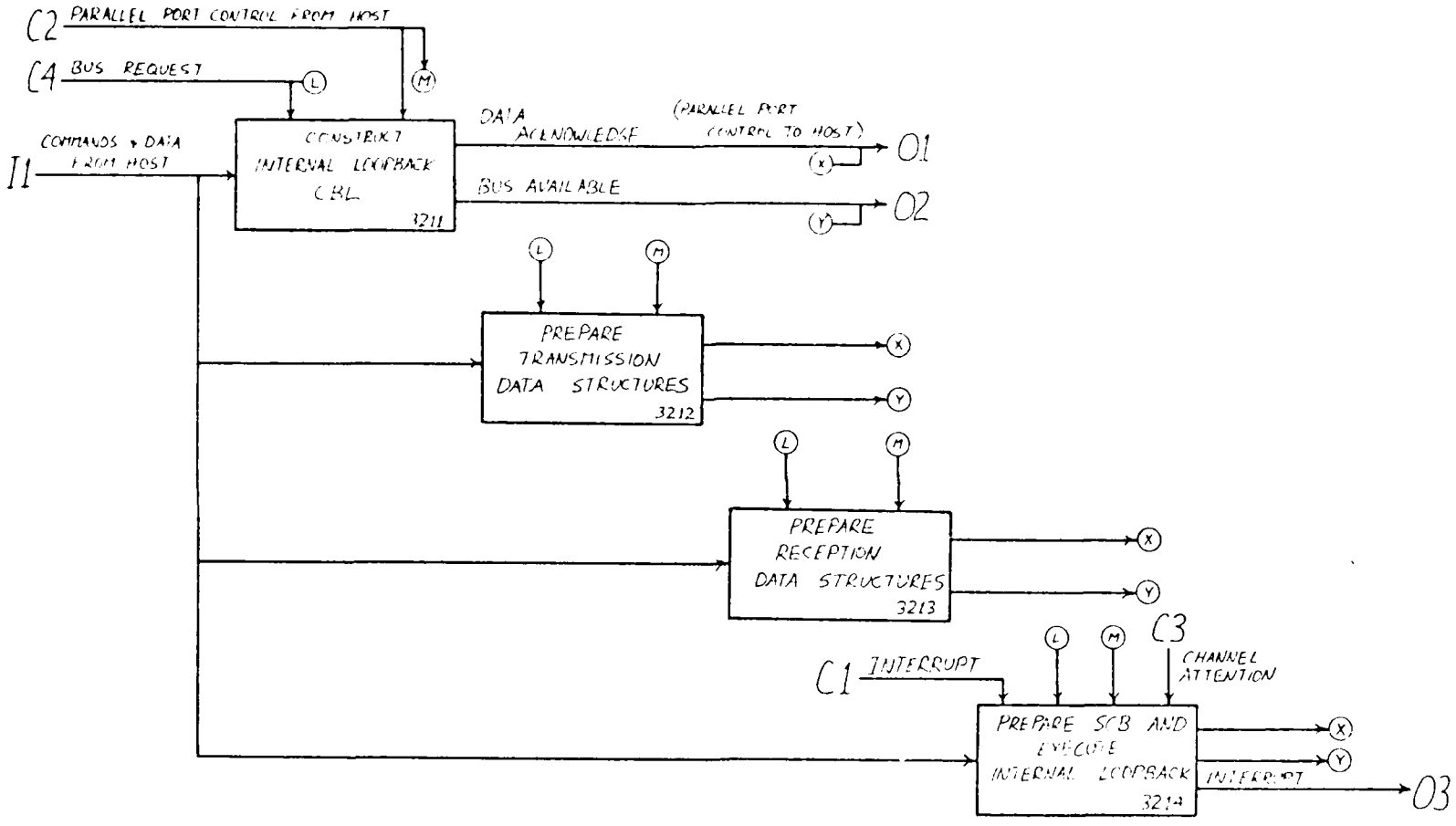
Perform Internal Loopback Set Up And Execution requires the host to create in NIB memory a Command Block List (CBL) containing the Configure command with internal loopback specified, and the Transmit command. Since a data packet is transmitted and received (although internal to the 82586), the transmission and reception data structures must also be prepared by the host. Finally, the 82586 System Control Block (SCB) is then prepared so that, when channel attention is applied, the 82586 knows to execute the internal loopback test. Execution of A3211, A3212, and A3213 can be in any order, but A3214 must occur last. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Prepare SCB And Execute Internal Loopback, the 82586 does take control as it executes the command. However, the host is not asking for the bus at this time, so there is no contention. In the other three activities, the 82586 does not ever take control of the bus.

A3211 Construct Internal Loopback CBL handles the creation of a CBL containing the Configure command set for internal loopback. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements while accepting commands and data.

A3212 Prepare Transmission Data Structures handles the initialization of the 82586 transmission data structures in anticipation of transmitting a data packet. The description of the inputs and outputs is the same as for A3211.

A3213 Prepare Reception Data Structures handles the initialization of the 82586 reception data structures in anticipation of receiving a data packet. The description of the inputs and outputs is the same as for A3211.

A3214 Prepare SCB And Execute Internal Loopback handles the preparation of the SCB and the execution of the Configure command that is set for internal loopback. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The host must also acknowledge the interrupt (shown as a control input) that was generated by the NIB upon completion of a previous activity. The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with command execution. The interrupt output from this activity is typically the result of the host setting the interrupt bit in the command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host.



NODE: A321 PERFORM INTERNAL LOOPBACK SET UP AND EXECUTION

A3211 Construct Internal Loopback CBL

Abstract: This diagram decomposes the activity Construct Internal Loopback CBL into its major functions.

Construct Internal Loopback CBL requires the host to create in NIB memory a Command Block List (CBL) containing the Configure command set for internal loopback, and a Transmit command for the packet to be internally passed. Execution of activities A32111 and A32112 is sequential as implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Construct Internal Loopback CBL, the 82586 does not ever take control of the bus, so there is no contention problem.

A32111 (Identical to activity A211)

A32112 Load Internal Loopback Configure And Transmit Commands In CBL loads the Configure and Transmit parameters shown as inputs into the CBL located by activity A32111. After requesting the bus, the host loads the Configure and Transmit command parameters (according to their respective data structures) into the CBL via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.

Again, the order of execution is A32111, A32112.

A3311 Construct External Loopback CBL

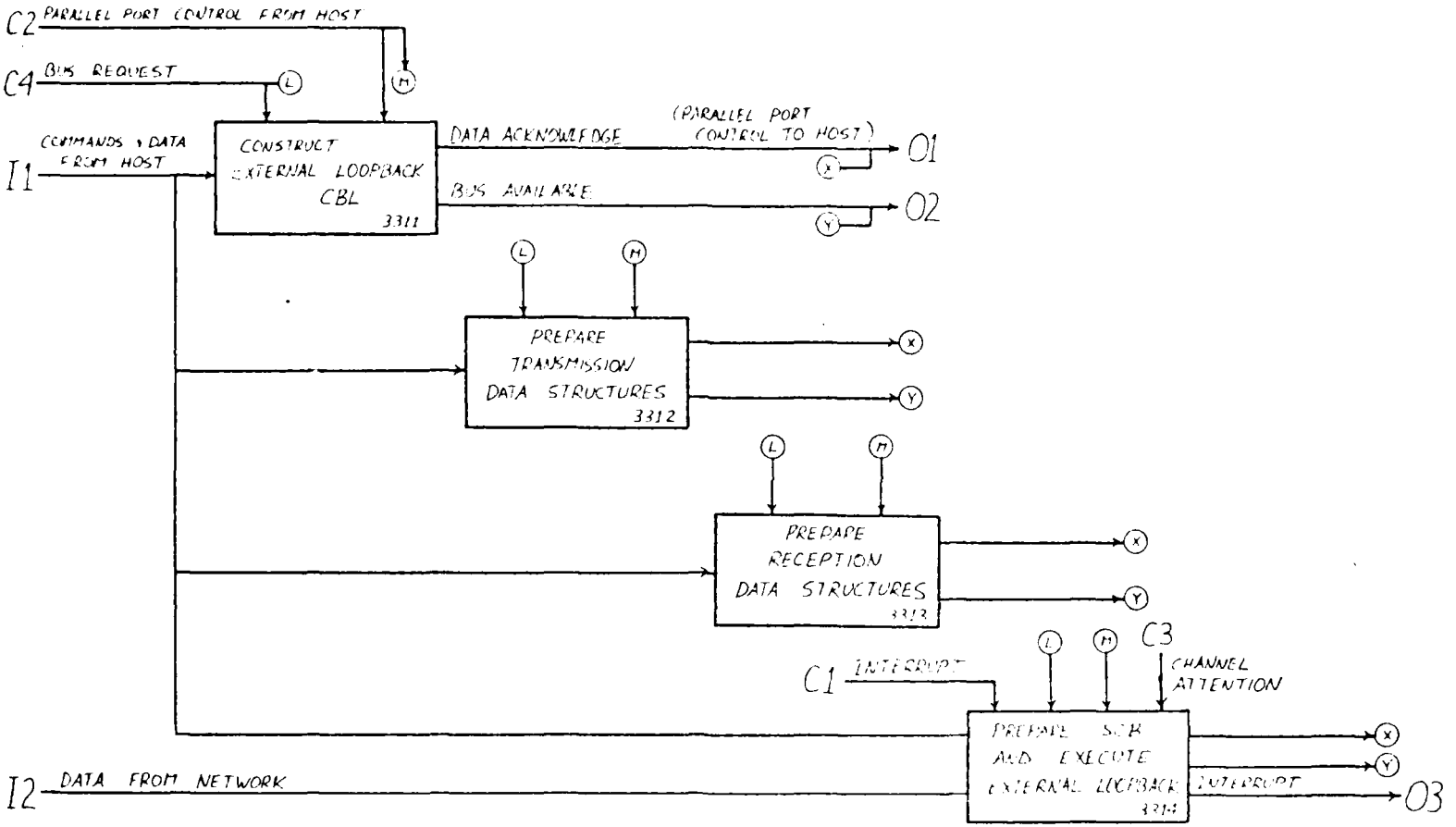
Abstract: This diagram decomposes the activity Construct External Loopback CBL into its major functions.

Construct External Loopback CBL requires the host to create in NIB memory a Command Block List (CBL) containing the Configure command set for external loopback, and a Transmit command for the packet to be externally transmitted. Execution of activities A33111 and A33112 is sequential as implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Construct External Loopback CBL, the 82586 does not ever take control of the bus, so there is no contention problem.

A33111 (Identical to activity A211)

A33112 Load External Loopback Configure And Transmit Commands In CBL loads the Configure and Transmit parameters shown as inputs into the CBL located by activity A33111. After requesting the bus, the host loads the Configure and Transmit command parameters (according to their respective data structures) into the CBL via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.

Again, the order of execution is A33111, A33112.



NODE: A331 PERFORM EXTERNAL LOOPBACK SET UP AND EXECUTION

A331 Perform External Loopback Set Up And Execution

Abstract: This diagram decomposes the activity Perform External Loopback Set Up And Execution into its major functions.

Perform External Loopback Set Up And Execution requires the host to create in NIB memory a Command Block List (CBL) containing the Configure command with enternal loopback specified, and a Transmit command. Since a data packet is transmitted and received, the transmission and reception data structures must also be prepared by the host. Finally, the 82586 System Control Block (SCB) is then prepared so that, when channel attention is applied, the 82586 knows to execute the external loopback test. Execution of A3311, A3312, and A3313 can be in any order, but A3314 must be last. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity A3314, the 82586 does take control as it executes the commands. However, the host is not actively asking for the bus at these times, so there is no contention. In the other three activities, the 82586 does not ever take control of the bus.

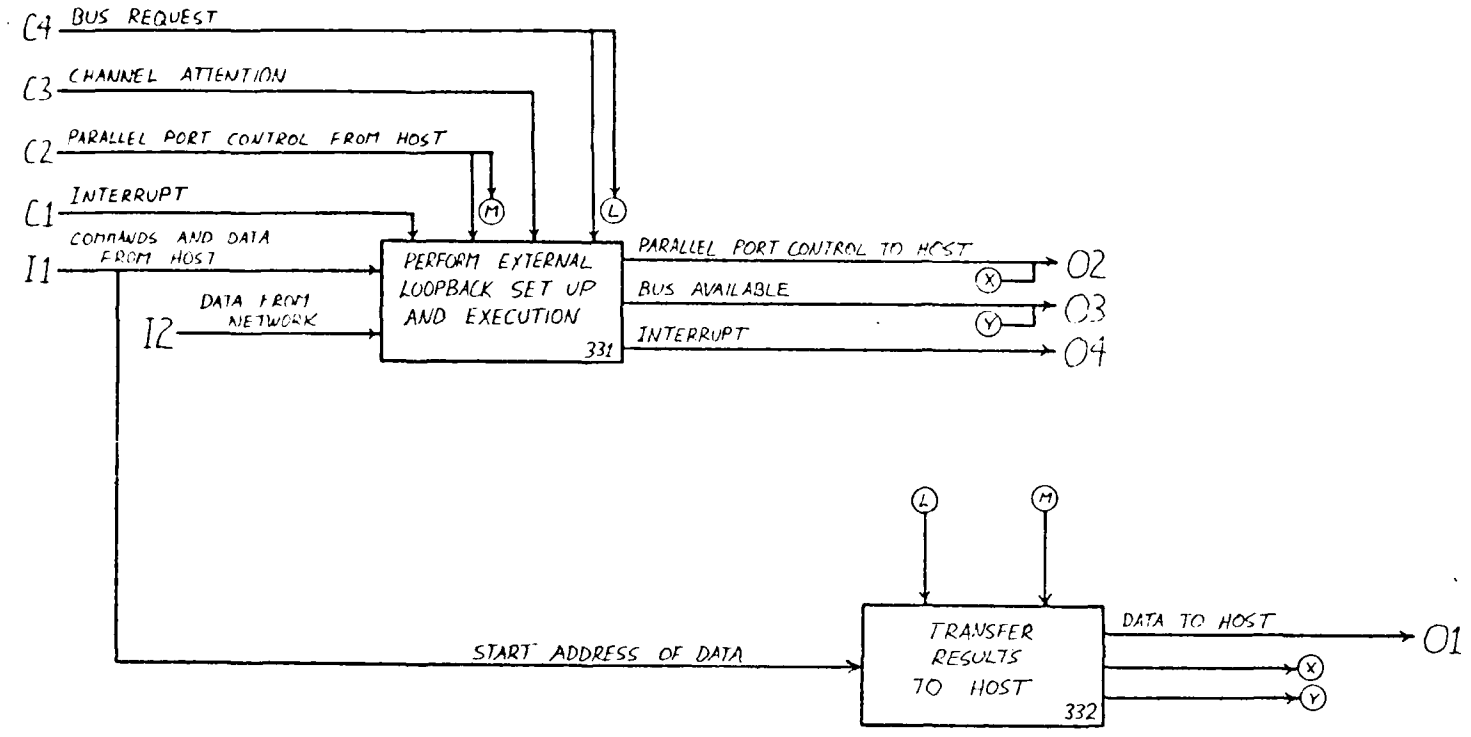
A3311 Construct External Loopback CBL handles the creation of a CBL containing the Configure command (for external loopback) and the Transmit command. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements while accepting commands and data.

A3312 (Identical to activity A3212)

A3313 (Identical to activity A3213)

A3314 Prepare SCB And Execute External Loopback handles the preparation of the SCB and the execution of the Configure and Transmit commands for external loopback. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The host must also acknowledge the interrupt (shown as a control input) that was generated by the NIB upon completion of a previous activity (such as A32143). The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with command execution. During execution the data transmitted is received from the network. The interrupt output from this activity is typically the result of the host setting the interrupt bit in the command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host before the next activity prompted by channel attention can be completed.

Again, the execution of activities A3311, A3312, and A3313 can be in any order, but A3314 must be last.



NODE: A33 PERFORM EXTERNAL LOOPBACK

A33 Perform External Loopback

Abstract: This diagram decomposes the activity Perform External Loopback into its major functions.

Perform External Loopback is composed of activities which set up and execute the 82586 external loopback test and transfer the results to the host. The order of execution of activities A331 and A332 is sequential as implied by their positioning from upper left to lower right.

A331 Perform External Loopback Set Up And Execution prepares the NIB for the external loopback test and then executes it. After requesting the bus, the host uses parallel port controls to input commands and data for preparing and executing this activity. The host must also acknowledge the interrupt (shown as a control input) that was generated by the NIB upon completion of a previous activity. The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with command execution. During execution, a data packet is transmitted and received, so data from the network is shown as an input. The bus available output from this activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Perform External Loopback Set Up And Execution, the 82586 does take control of the bus during the execution of the commands. However, the host is not actively asking forthe bus at these times, so there is no contention. The interrpt output from this activity is typically the result of the host setting the interrupt bit in the command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host before the next activity prompted by a channel attention can be completed.

A332 (Identical to activity A312)

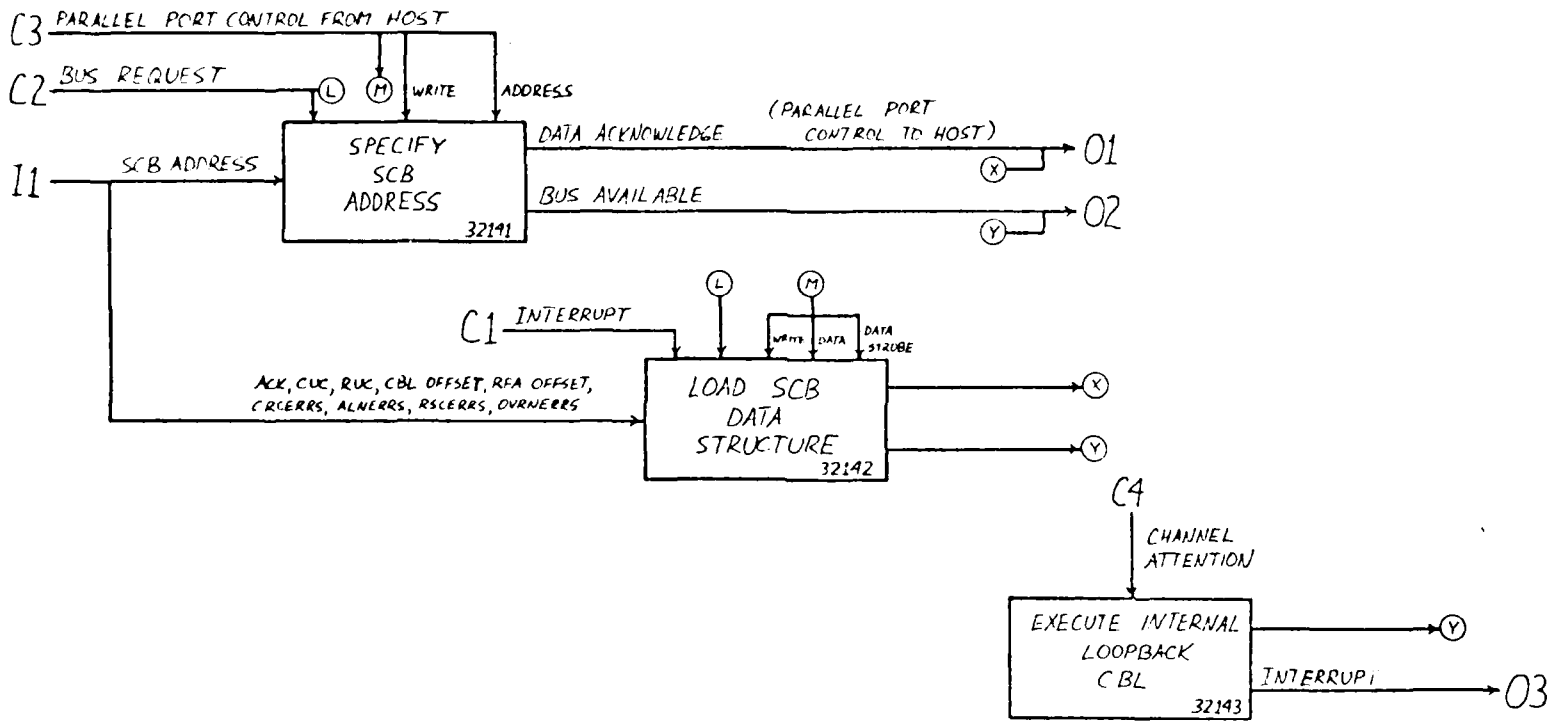
Again, the execution of these activities must be in the order A331, A332.

SAME AS NODE A312

NODE: A322 TRANSFER RESULTS TO HOST

A322 Transfer Results To Host

Abstract: This diagram, other than the content of the data transmitted, is identical in function to activity A312. This activity transmits the results of the internal loopback test, whereas activity A312 transmits the results of the diagnose test.



NODE: A3214 PREPARE SCB AND EXECUTE INTERNAL LOOPBACK

A3214 Prepare SCB And Execute Internal Loopback

Abstract: This diagram decomposes the activity Prepare SCB And Execute Internal Loopback into its major functions.

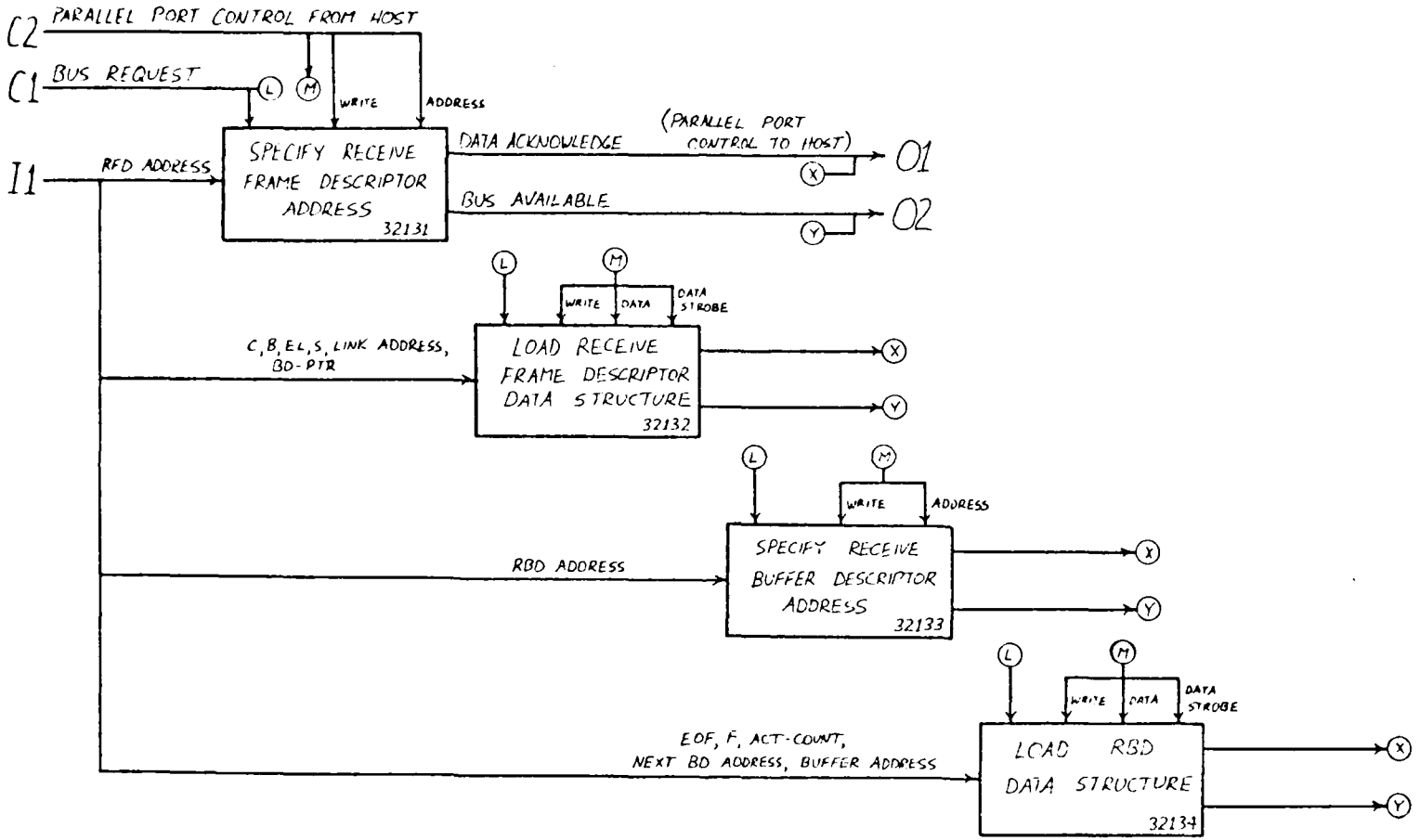
Prepare SCB And Execute Internal Loopback requires the host to prepare the 82586 System Control Block (SCB) so that, when channel attention is applied, the 82586 knows to execute the Configure command set for the internal loopback test. Execution of A32141, A32142, and A32143 is sequential and is implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activities A32141 and A32142, the 82586 does not ever take control of the bus, so there is no contention for the bus. During activity A32143, the 82586 does take control during execution of the command. However, the host is not actively asking for the bus during this time, so there is again no contention.

A32141 (Identical to activity A213)

A32142 Load SCB Data Structure loads the SCB parameters shown into the SCB data structure at the address specified in A32141. Note that this activity differs from activity A214 in that the parameter RUC must be included to turn the 82586 Receive Unit (RU) on. After requesting the bus, the host loads this data into the SCB data structure via the parallel port controls write, data, and data strobe. Note that the interrupt from a previous activity (such as A3115) must be acknowledged in the input ACK. The 82586 responds to this activity with acknowledgements.

A32143 Execute Internal Loopback CBL is the execution of the Configure (set for internal loopback) and Transmit commands in the CBL by the 82586. It is begun by the host applying channel attention to the NIB. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the Transmit command).

Again, the execution of these activities must be in the order A32141, A32142, A32143.



NODE: A3213 PREPARE RECEPTION DATA STRUCTURES

A3213 Prepare Reception Data Structures

Abstract: This diagram decomposes the activity Prepare Reception Data Structures into its major functions.

Prepare Reception Data Structures handles the preparation of the Receive Frame Descriptor (RFD) and the Receive Buffer Descriptor (RBD) for an incoming data packet. Execution of A32131, A32132, A32133, and A32134 is sequential as implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Prepare Reception Data Structures, the 82586 does not ever take control, so there is no contention for the bus.

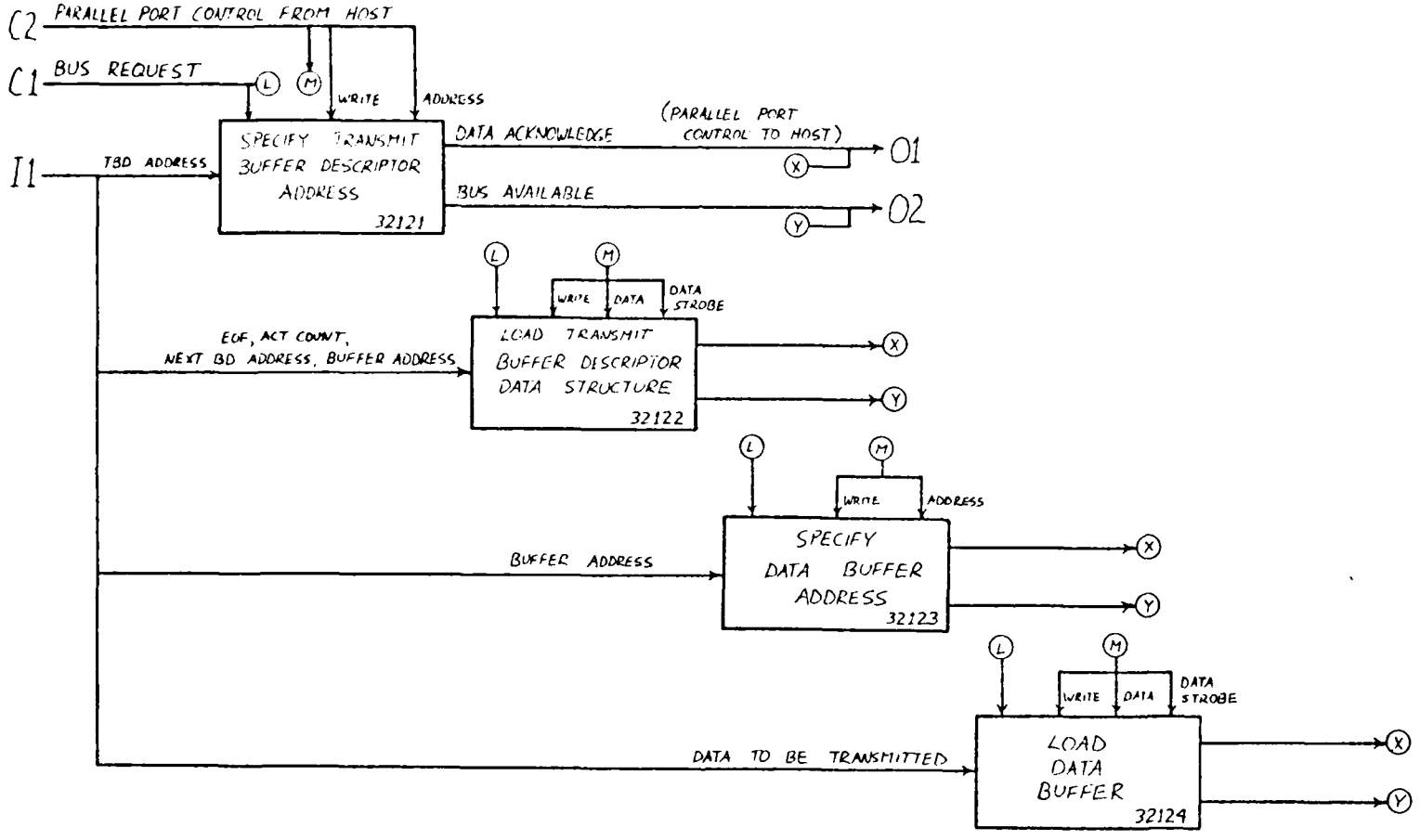
A32131 Specify Receive Frame Descriptor Address is performed by the host prior to loading data into the RFD. After requesting the bus, the host applies the RFD address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

A32132 Load Receive Frame Descriptor Data Structure loads the RFD parameters shown as inputs into the RFD located by activity A32131. After requesting the bus, the host loads the RFD parameters (according to the RFD data structure) into the RFD via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.

A32133 Specify Receive Buffer Descriptor Address is performed by the host prior to loading data into the RBD. After requesting the bus, the host applies the RBD address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

A32134 Load RBD Data Structure loads the RBD parameters shown as inputs into the RBD located by activity A32133. After requesting the bus, the host loads the RBD parameters (according to the RBD data structure) into the RBD via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.

Again, the execution of activities is in the order A32131, A32132, A32133, and A32134.



NODE: A3212 PREPARE TRANSMISSION DATA STRUCTURES

A3212 Prepare Transmission Data Structures

Abstract: This diagram decomposes the activity Prepare Transmission Data Structures into its major functions.

Prepare Transmission Data Structures handles the preparation of the Transmit Buffer Descriptor (TBD) and the Data Buffer containing the data to be transmitted. Execution of A32121, A32122, A32123, and A32124 is sequential as implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Prepare Transmission Data Structures, the 82586 does not ever take control, so there is no contention for the bus.

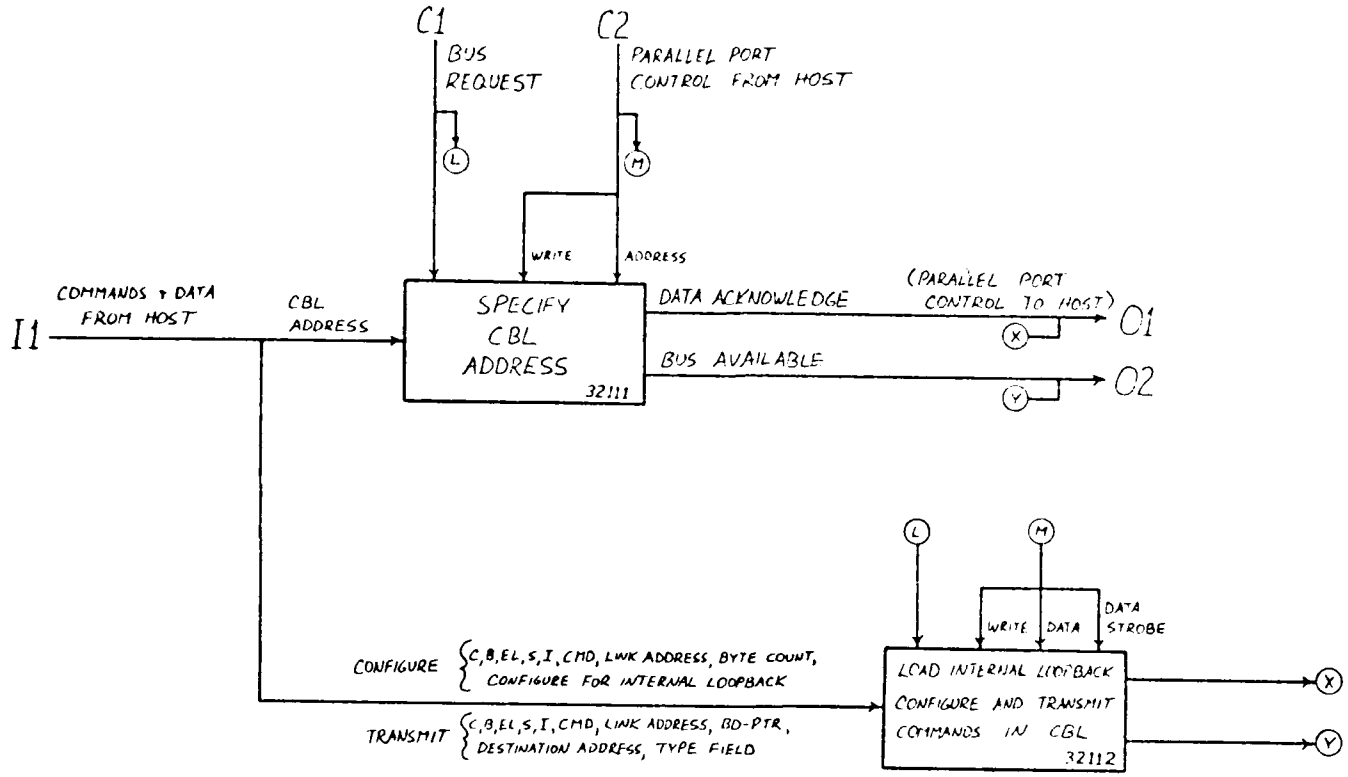
A32121 Specify Transmit Buffer Descriptor Address is performed by the host prior to loading data into the TBD. After requesting the bus, the host applies the TBD address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

A32122 Load Transmit Buffer Descriptor Data Structure loads the TBD parameters shown as inputs into the TBD located by activity A32121. After requesting the bus, the host loads the TBD parameters (according to the TBD data structure) into the TBD via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.

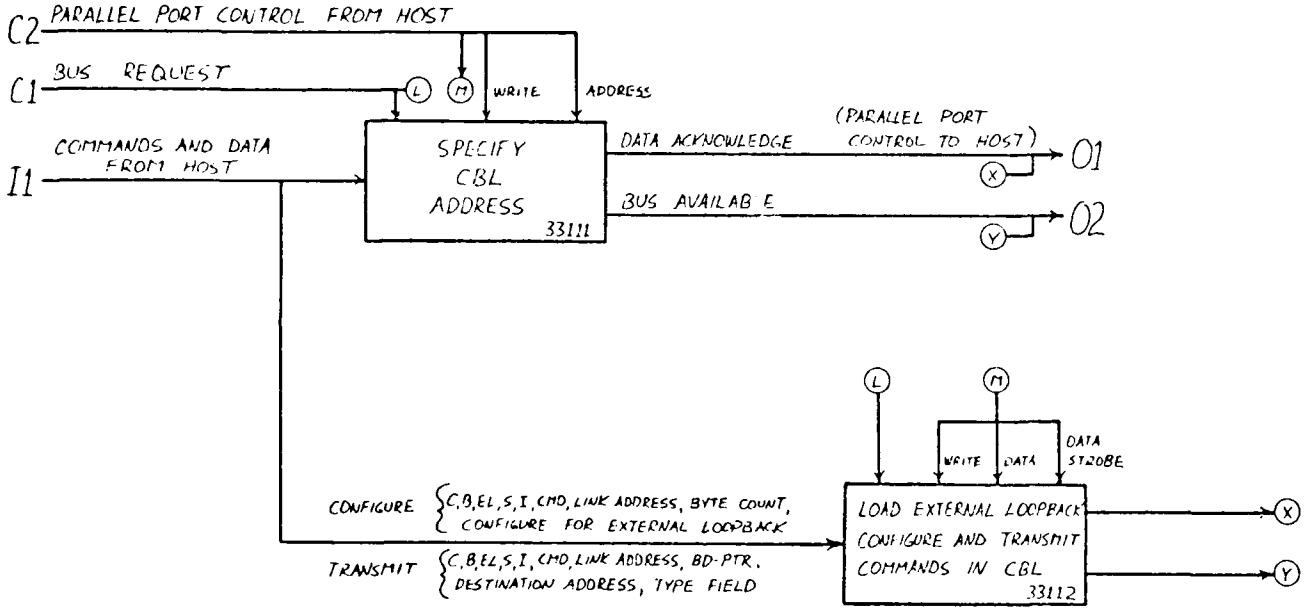
A32123 Specify Data Buffer Address is performed by the host prior to loading the data to be transmitted into the buffer. After requesting the bus, the host applies the data buffer address to the parallel port via the write and address controls. The 82586 responds with data acknowledge.

A32124 Load Data Buffer loads the data buffer with the data to be transmitted. After requesting the bus, the host loads the buffer via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.j

Again, the execution of activities is in the order A32121, A32122, A32123, and A32124.



NODE: A3211 CONSTRUCT INTERNAL LOOPBACK CBL



NODE: A3311 CONSTRUCT EXTERNAL LOOPBACK CBL

A3312 Prepare Transmission Data Structures

Abstract: This activity performs the same function as activity A3212.

SAME AS NODE A3212

NODE: A3312 PREPARE TRANSMISSION DATA STRUCTURES

A3313 Prepare Reception Data Structures

Abstract: This activity performs the same function as activity A3213.

SAME AS NODE A3213

NODE: A3313 PREPARE RECEPTION DATA STRUCTURES

A3314 Prepare SCB And Execute External Loopback

Abstract: This diagram decomposes the activity Prepare SCB And Execute External Loopback into its major functions.

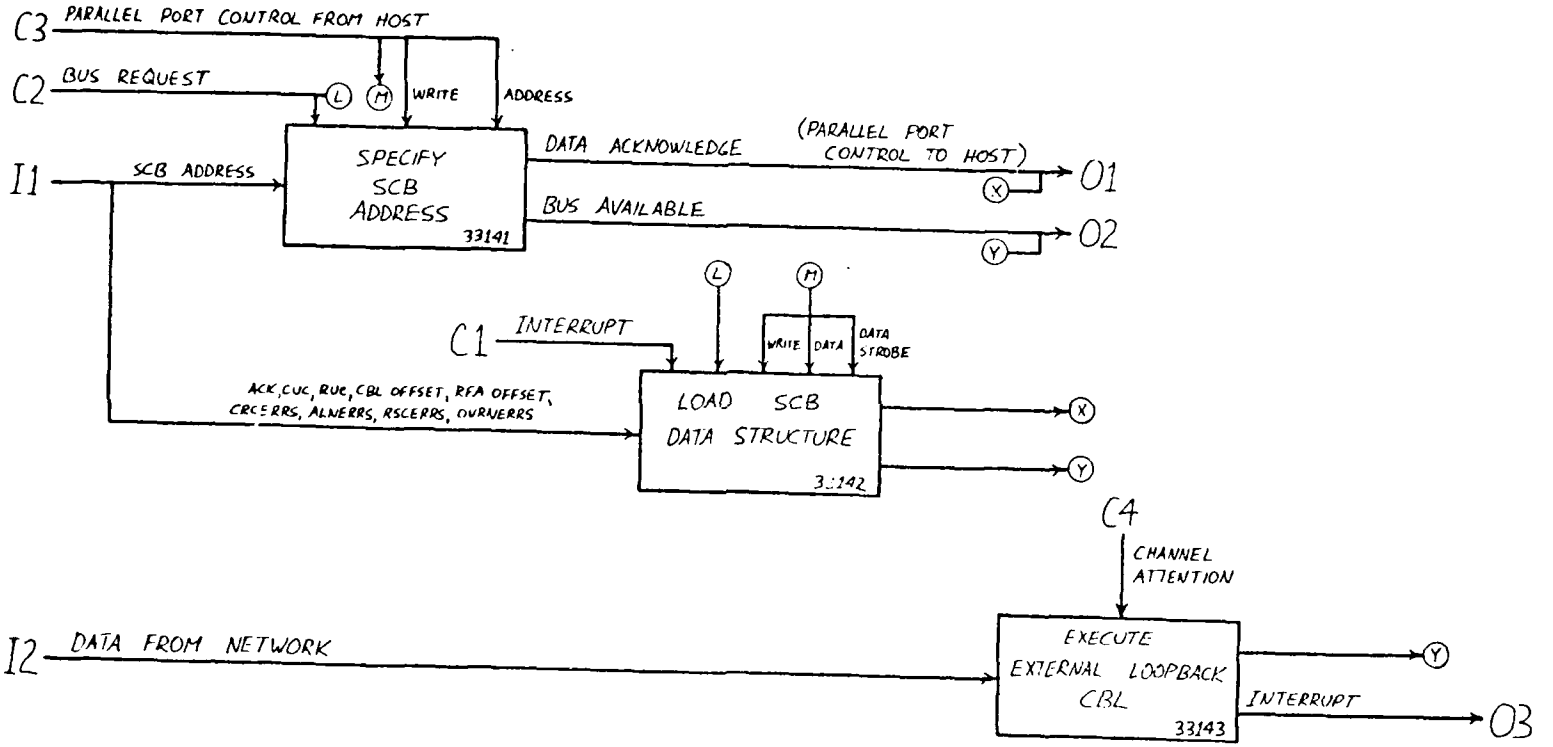
Prepare SCB And Execute External Loopback requires the host to prepare the 82586 System Control Block (SCB) so that, when channel attention is applied, the 82586 knows to execute the Configure command (set to external loopback) and the Transmit command. Execution of A33141, A33142, and A33143 is sequential and is implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activities A33141 and A33142, the 82586 does not ever take control of the bus, so there is no contention for the bus. During activity A33143, the 82586 does take control during execution of the command. However, the host is not actively asking for the bus during this time, so there is again no contention.

A33141 (Identical to activity A213)

A33142 (Identical to activity A32142)

A33143 Execute External Loopback CBL is the execution of the Configure (set for external loopback) and Transmit commands in the CBL by the 82586. It is begun by the host applying channel attention to the NIB. During execution, a data packet is transmitted and received. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the Transmit command).

Again, the execution of these activities must be in the order A33141, A33142, A33143.



NODE: A3314 PREPARE SCB AND EXECUTE EXTERNAL LOOPBACK

A332 Transfer Results To Host

Abstract: This activity performs the same function as activity A312.

SAME AS NODE A312

NODE: A332 TRANSFER RESULTS TO HOST

A34 Perform TDR Test

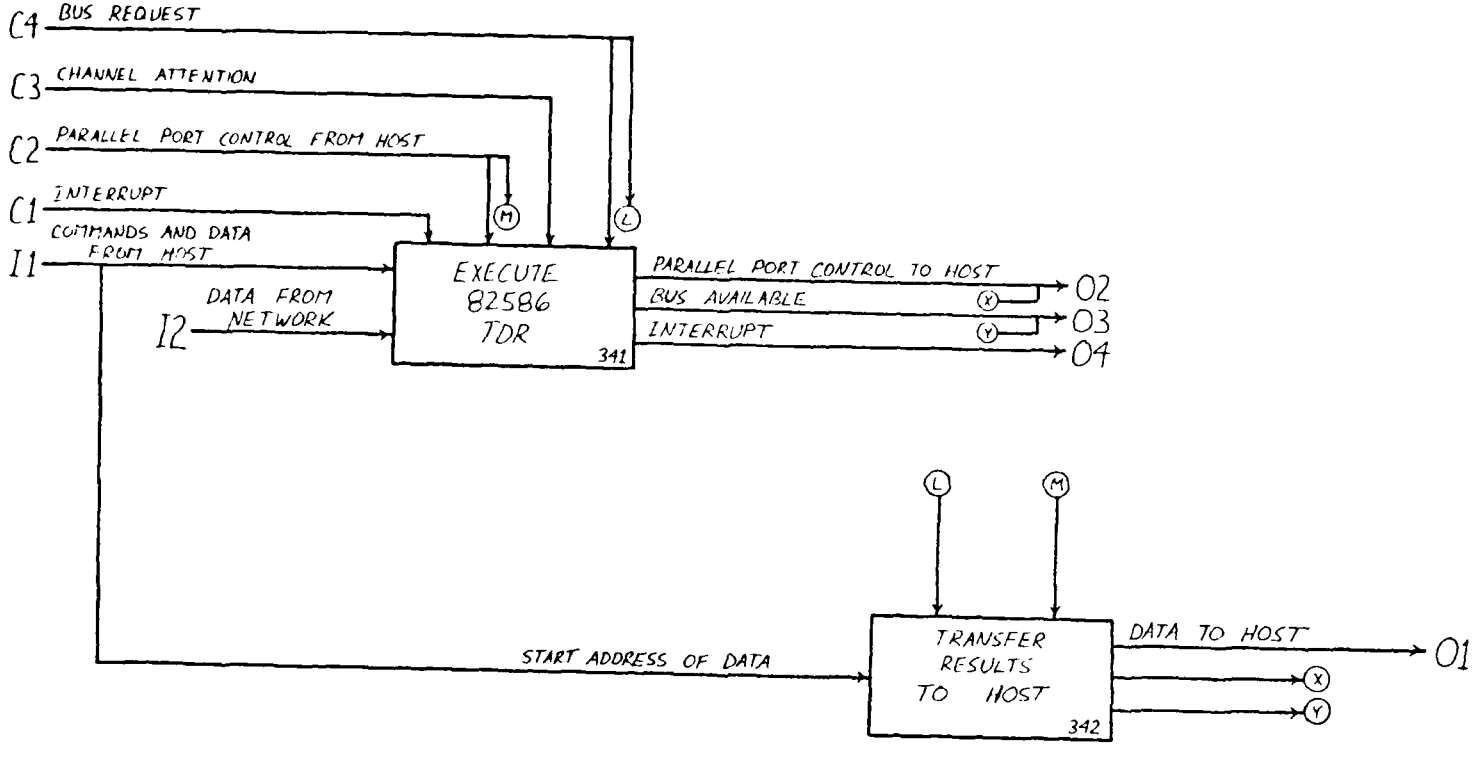
Abstract: This diagram decomposes the activity Perform TDR Test into its major functions.

Perform TDR Test is composed of activities which set up and execute the 82586 TDR command and transfer the results to the host. The order of execution of activities A341 and A342 is sequential as implied by their positioning from upper left to lower right.

A341 Execute 82586 TDR performs the Time Domain Reflectometer test. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The host must also acknowledge the interrupt (shown as a control input) that was generated by the NIB upon completion of a previous activity (such as A33143). The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with command execution. The bus available output from this activity is an indication as to whther or not the 82586 has taken control of the NIB bus. During the activity Execute 82586 TDR, the 82586 does take control of the bus during the execution of the command. However, the host is not actively asking for the bus at this time, so there is no contention. The interrupt output from this activity is typically the result of the host setting the interrupt bit in the command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host before the next activity prompted by a channel attention can be completed.

A342 (Identical to activity A312)

Again, the execution of these activities must be in the order A341, A342.



NODE: A34 PERFORM TDR TEST

A341 Execute 82586 TDR

Abstract: This diagram decomposes the activity Execute 82586 TDR into its major functions.

Execute 82586 TDR requires the host to create in NIB memory a Command Block List (CBL) containing the TDR command. The 82586 System Control Block (SCB) is then prepared so that, when channel attention is applied, the 82586 knows to execute the TDR command. Execution of A3411, A3412, A3413, A3414, and A3415 is sequential as implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Execute 82586 TDR, the 82586 does take control as it executes the command. However, the host is not actively asking for the bus at this time, so there is no contention.

A3411 (Identical to activity A211)

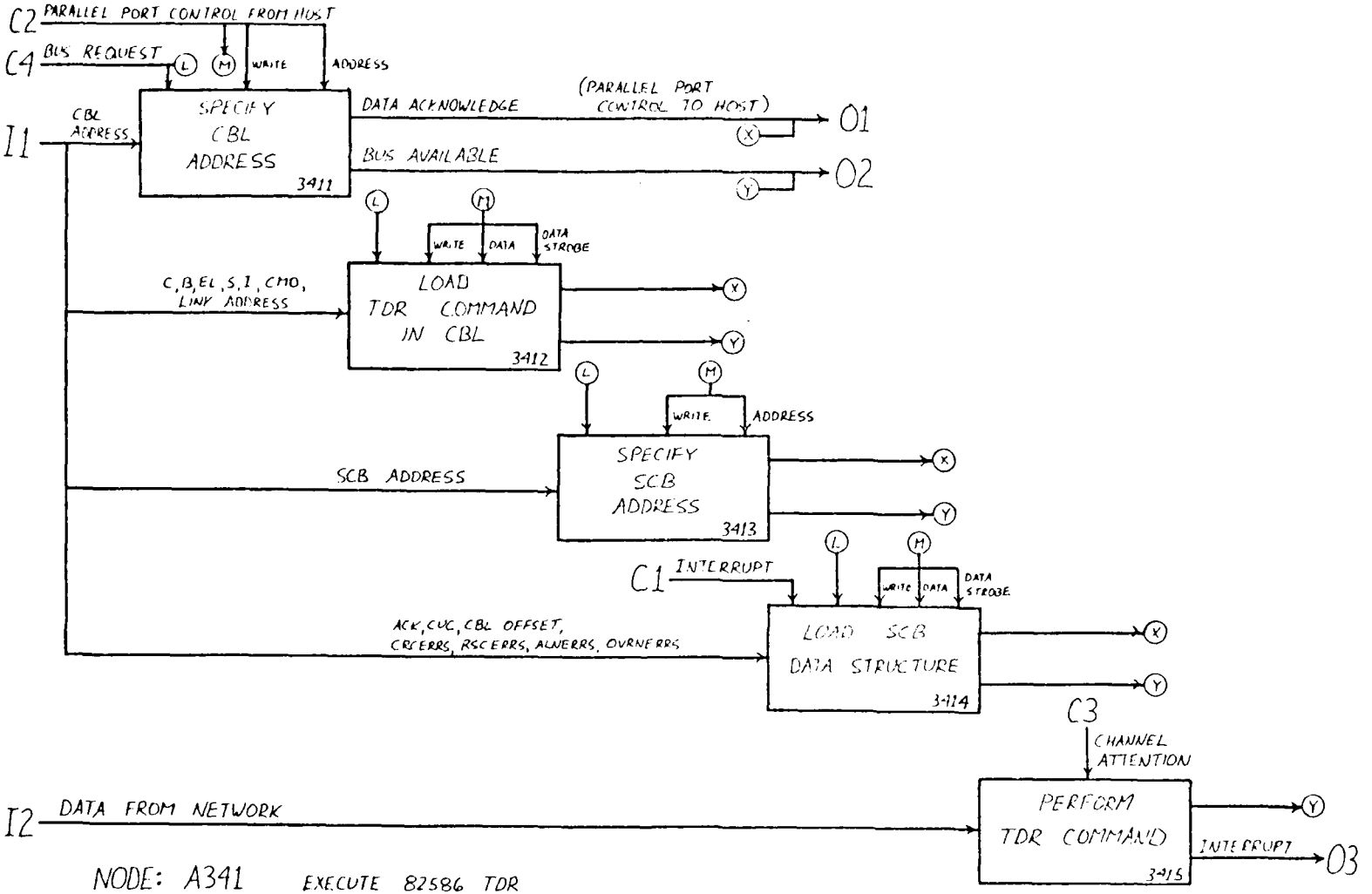
A3412 Load TDR Command In CBL loads the TDR parameters shown as inputs into the CBL located by activity A3411. After requesting the bus, the host loads the TDR command parameters (according to the TDR command data structure) into the CBL via the parallel port controls write, data, and data strobe. The 82586 responds with data acknowledgements.

A3413 (Identical to activity A213)

A3414 Load SCB Data Structure loads the SCB parameters shoen into the SCB data structure at the address specified in A3413. After requesting the bus, the host loads this data into the SCB data structure via the parallel port controls write, data, and data strobe. Note that the interrupt generated from a previous activity must be acknowledged in the input ACK. The 82586 responds to this activity with data acknowledgements.

A3415 Perform TDR Command is the execution of the TDR command by the 82586. It is begun by the host applying channel attention to the NIB, and the network response is collected by the NIB. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the TDR CBL).

Again, the execution of these activities must be in the order A3411, A3412, A3413, A3414, A3415.


C2
PARALLEL PORT CONTROL FROM HOST
C4
BUS REQUEST
L
M
WRITE
ADDRESS
I1
CBL ADDRESS
SPECIFY CBL ADDRESS
3411
DATA ACKNOWLEDGE
(PARALLEL PORT CONTROL TO HOST)
O1
BUS AVAILABLE
O2
X
Y
WRITE
DATA
DATA STROBE
C,B,EL,S,I,CMD, LINK ADDRESS
LOAD TDR COMMAND IN CBL
3412
SCB ADDRESS
SPECIFY SCB ADDRESS
3413
C1
INTERRUPT
ACK,CUC,CBL OFFSET, CRCERRS, RSCERRS, ALNERRS, OVRNERRS
LOAD SCB DATA STRUCTURE
3414
C3
CHANNEL ATTENTION
I2
DATA FROM NETWORK
PERFORM TDR COMMAND
3415
INTERRUPT
O3
NODE: A341
EXECUTE 82586 TDR

A342 Transfer Results To Host

Abstract: This activity performs the same function as activity A312.

A42 Transmit Network Data

Abstract: This diagram decomposes the activity Transmit Network Data into its major functions.

Transmit Network Data performs the task of data transmission over the network. This is accomplished by constructing a Command Block List (CBL) containing one or more Transmit commands, preparing the transmission data structures, executing the command(s), and then verifying their correct transmission. Execution of activities A421, A422, A423, and A424 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the NIB bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from every activity.

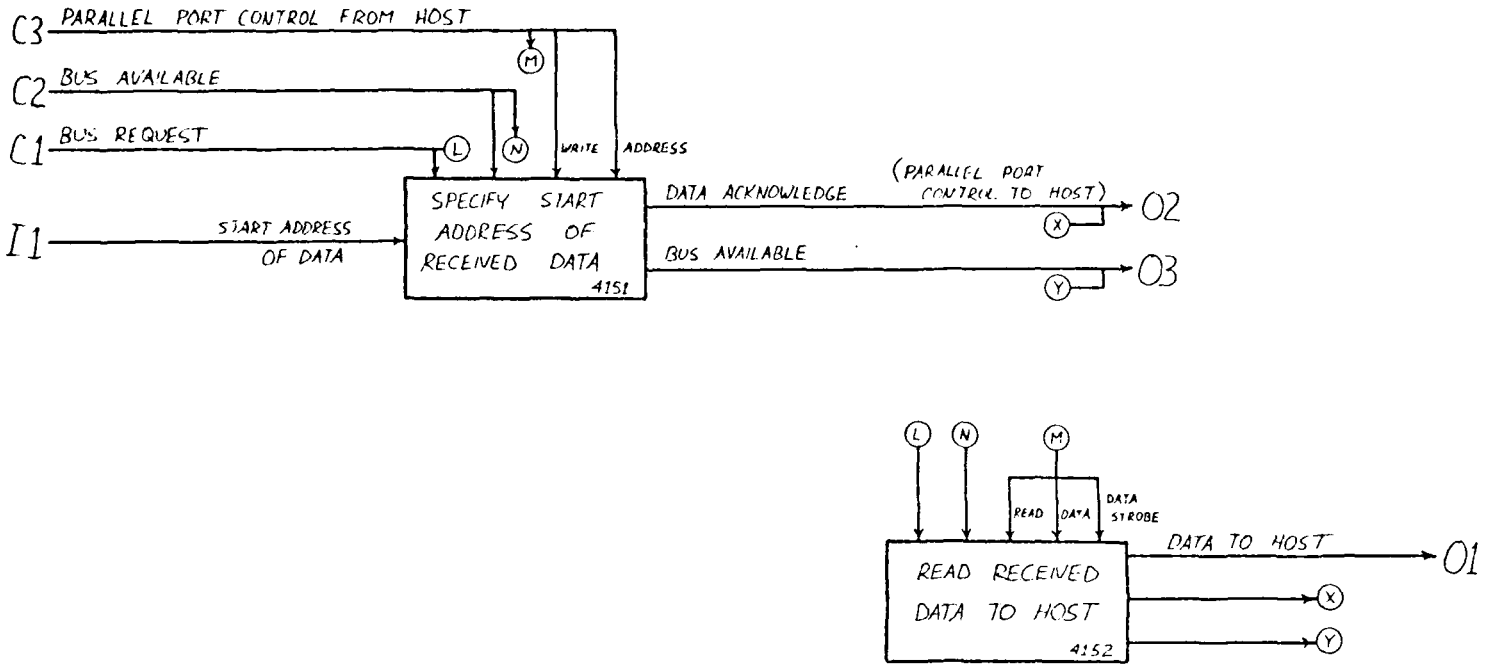
A421 Construct Transmit CBL prepares a CBL of one or more Transmit commands. After requesting the available bus, the host uses parallel port controls to input commands and data from the host. The NIB responds with data acknowledges while accepting the commands and data.

A422 Prepare Transmission Data Structures performs the same function as activity A3212. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A423 Prepare SCB And Execute Transmit handles the set up of the SCB and the execution of the Transmit command CBL. After requesting the available bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements. The interrupt from a previous activity must be acknowledged by the host prior to executing the CBL. The host starts the transmission process by applying channel attention to the NIB, and the result is the transmission of data to the network. The NIB signals an interrupt upon completion of command execution.

A424 Transfer Transmit Command Status To Host transfers the status of the data transmission to the host. After requesting the available bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements. The result of this activity is the transfer of the Transmit command status word to the host.

Again, the order of execution is A421, A422, A423, A424.



NODE: A415 TRANSFER RECEIVED DATA TO HOST

A415 Transfer Received Data To Host

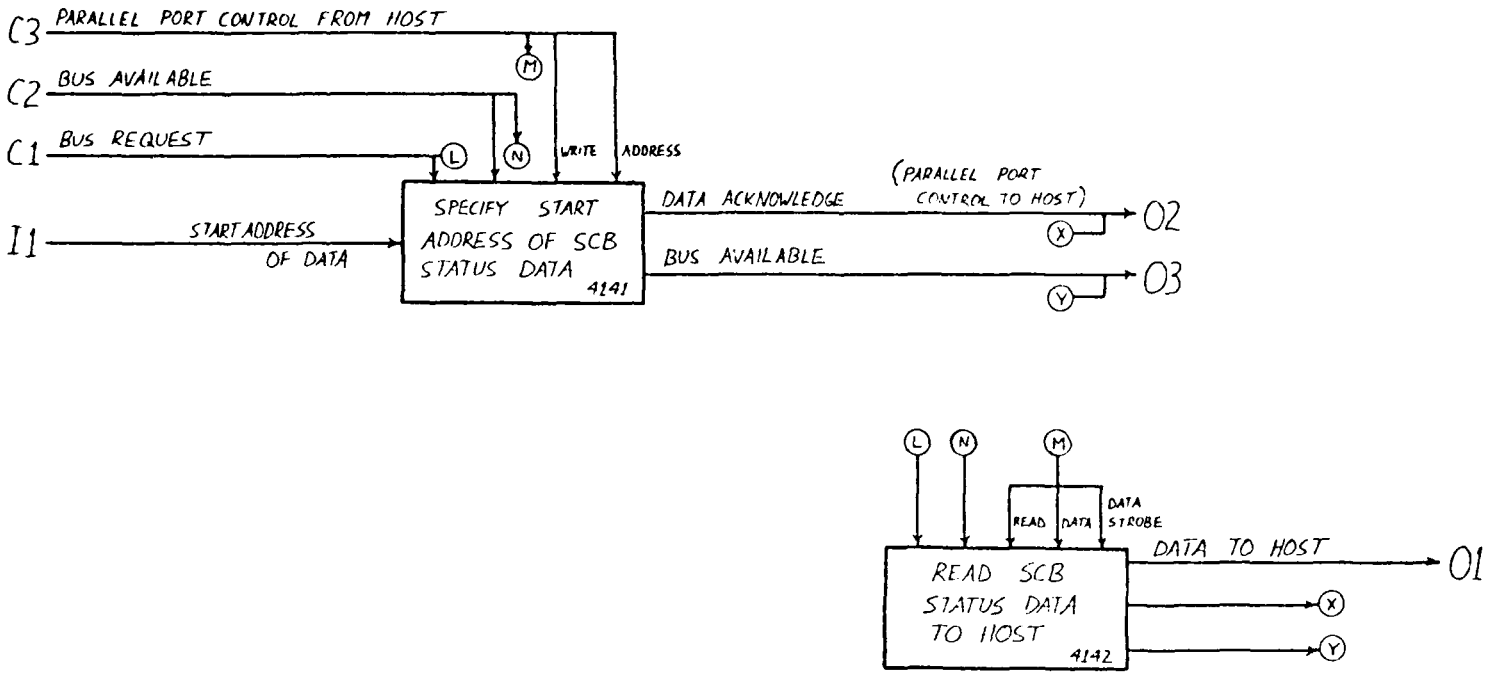
Abstract: This diagram decomposes the activity Transfer Received Data To Host into its major functions.

Transfer Received Data To Host handles the transfer of the data received from the network to the host. Execution of activities A4151 and A4152 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from activities A4151 and A4152.

A4151 Specify Start Address Of Received Data is performed by the host prior to transferring the received data to the host. After requesting the available bus, the host applies the start address of the received data to the parallel port via the write and address controls. The NIB responds with data acknowledge.

A4152 Read Received Data To Host handles the function of transferring the received data. After requesting the bus, the host uses parallel part controls read, data, and data strobe to step through NIB memory and read the status data. Data is transferred to the host, and the NIB responds with data acknowledgements.

Again, execution is in the order A4151, A4152.



NODE: A414 TRANSFER SCB STATUS TO HOST

A414 Transfer SCB Status To Host

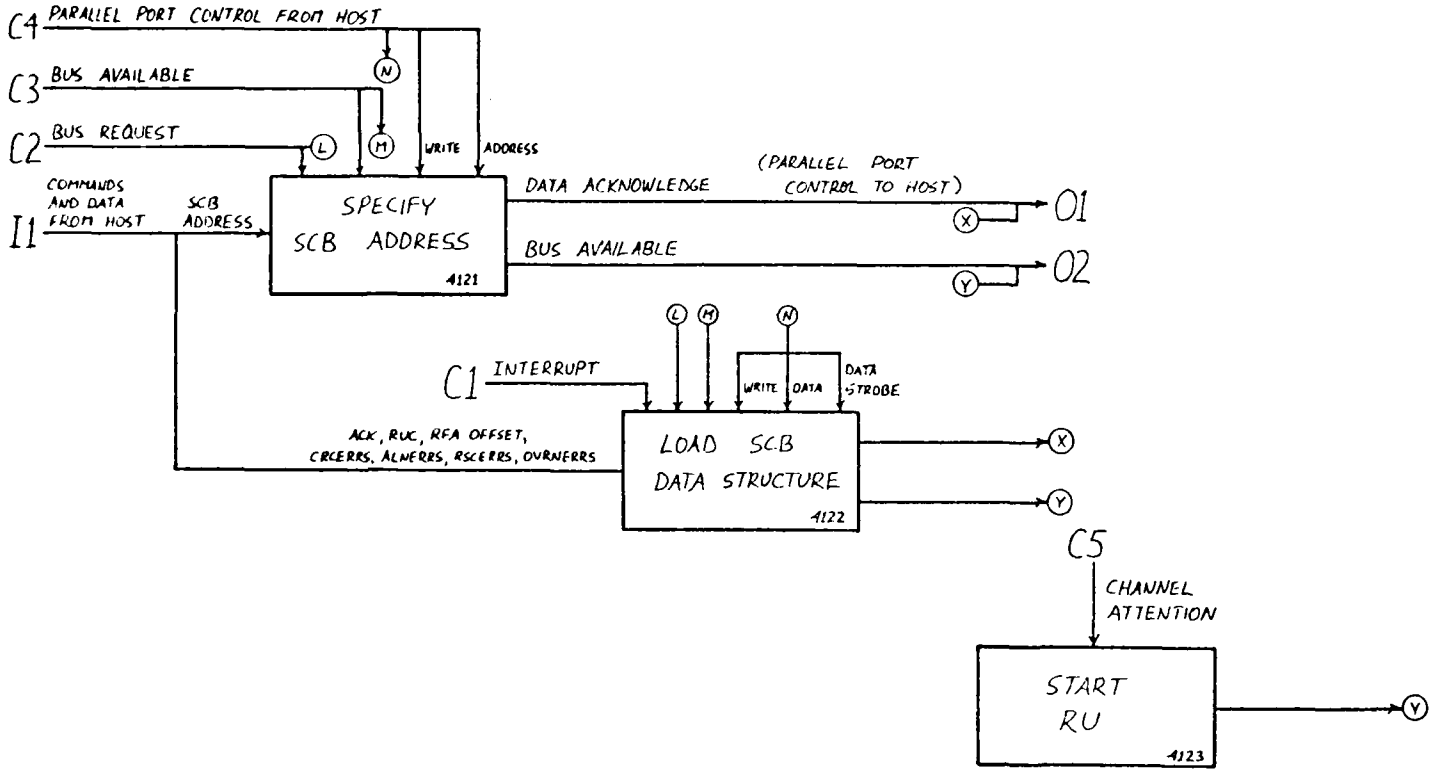
Abstract: This diagram decomposes the activity Transfer SCB Status To Host into its major functions.

Transfer SCB Status To Host handles the movement of the System Control Block (SCB) status word from NIB memory to host memory. Execution of A4141 and A4142 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from activities A4141 and A4142.

A4141 Specify Address Of SCB Status Data is performed by the host prior to transferring the SCB status word to the host. After requesting the available bus, the host applies the start address of the SCB status word to the parallel port via the write and address controls. The NIB responds with data acknowledge.

A4142 Read SCB Status Data To Host handles the function of transferring the status data. After requesting the bus, the host uses the parallel port controls read, data, and data strobe to step through NIB memory and read the status data. Data is transferred to the host, and the NIB responds with data acknowledgements.

Again, execution is in the order A4141, A4142.



NODE: A412 PREPARE SCB START RU

A412 Prepare SCB Start RU

Abstract: This diagram decomposes the activity Prepare SCB Start RU into its major functions.

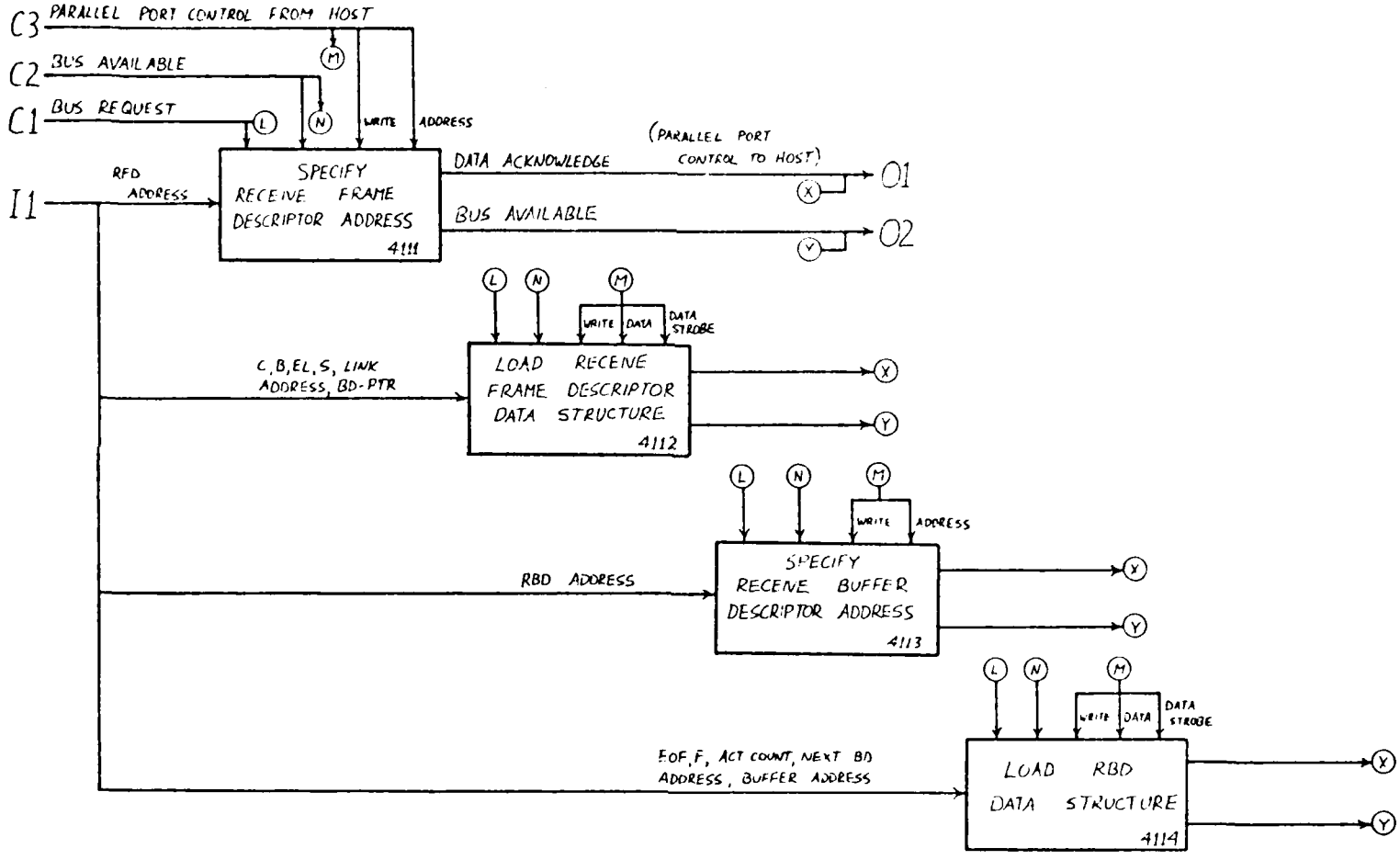
Prepare SCB Start RU enables the NIB to receive and store in NIB memory an incoming data packet. The command to start the Receive Unit (RU) of the 82586 is placed in the System Control Block (SCB), and channel attention is applied by the host to start the RU. Execution of activities A4121, A4122, and A4123 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from activities A4121 and A4122.

A4121 Specify SCB Address performs the same function as activity A213. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4122 Load SCB Data Structure performs the same function as activity A214. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4123 Start RU handles the starting of the 82586 RU. The RU is started by the host applying channel attention to the NIB, using the SCB prepared in activity A4122. The NIB does not control the bus during this activity.

Again, the order of execution is A4121, A4122, A4123.



NODE: A411 PREPARE RECEPTION DATA STRUCTURES

A411 Prepare Reception Data Structures

Abstract: This diagram decomposes the activity Prepare Reception Data Structures into its major functions.

Prepare Reception Data Structures Prepare Reception Data Structures handles the preparation of the Receive Frame Descriptor (RFD) and the Receive Buffer Descriptor (RBD) for an incoming data packet. Execution of A4111, A4112, A4113, and A4114 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the NIB bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from every activity.

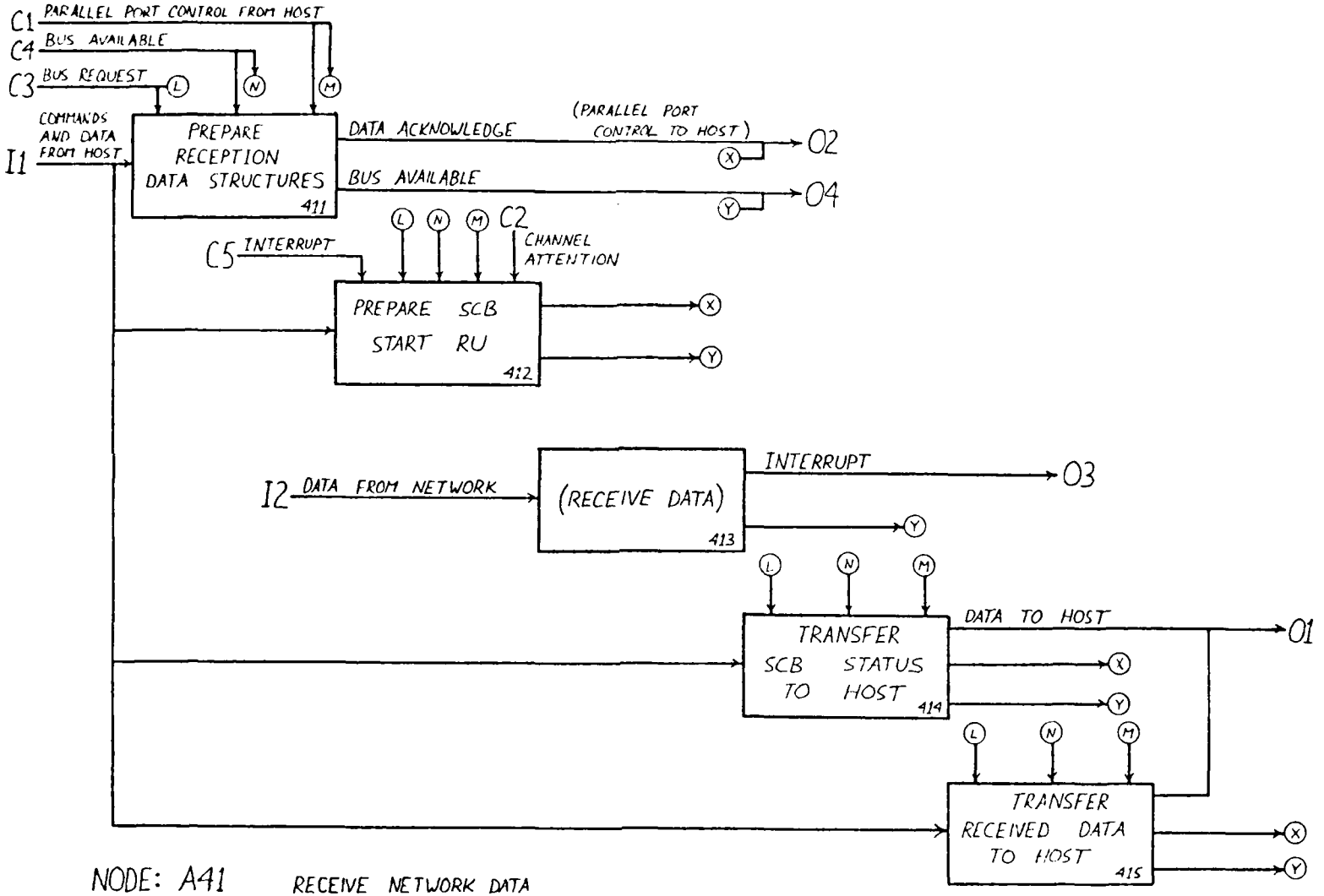
A4111 Specify Receive Frame Descriptor Address performs the same function as activity A32131. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4112 Load Receive Frame Descriptor Data Structure performs the same function as activity A32132. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4113 Specify Receive Buffer Descriptor Address performs the same function as activity A32133. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4114 Load RBD Data Structure performs the same function as activity A32134. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

Again, execution is in the order A4111, A4112, A4113, A4114.



NODE: A41 RECEIVE NETWORK DATA

A41 Receive Network Data

Abstract: This diagram decomposes the activity Receive Network Data into its major functions.

Receive Network Data is accomplished in three steps. First, the host must prepare the reception data structures and start the 82586 Receive Unit (RU). Second, data must be received. Third, the System Control Block (SCB) status word must be examined by the host to ensure correct reception. Activities A411, A412, A413, A414, and A415 accomplish these three functions, and their execution is sequential as implied by their positioning from upper left to lower right. Note, however, that A413 may preempt the execution of another activity since data reception has highest priority. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from activities A411, A412, A414, and A415.

A411 Prepare Reception Data Structures performs the same function as activity A3213. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

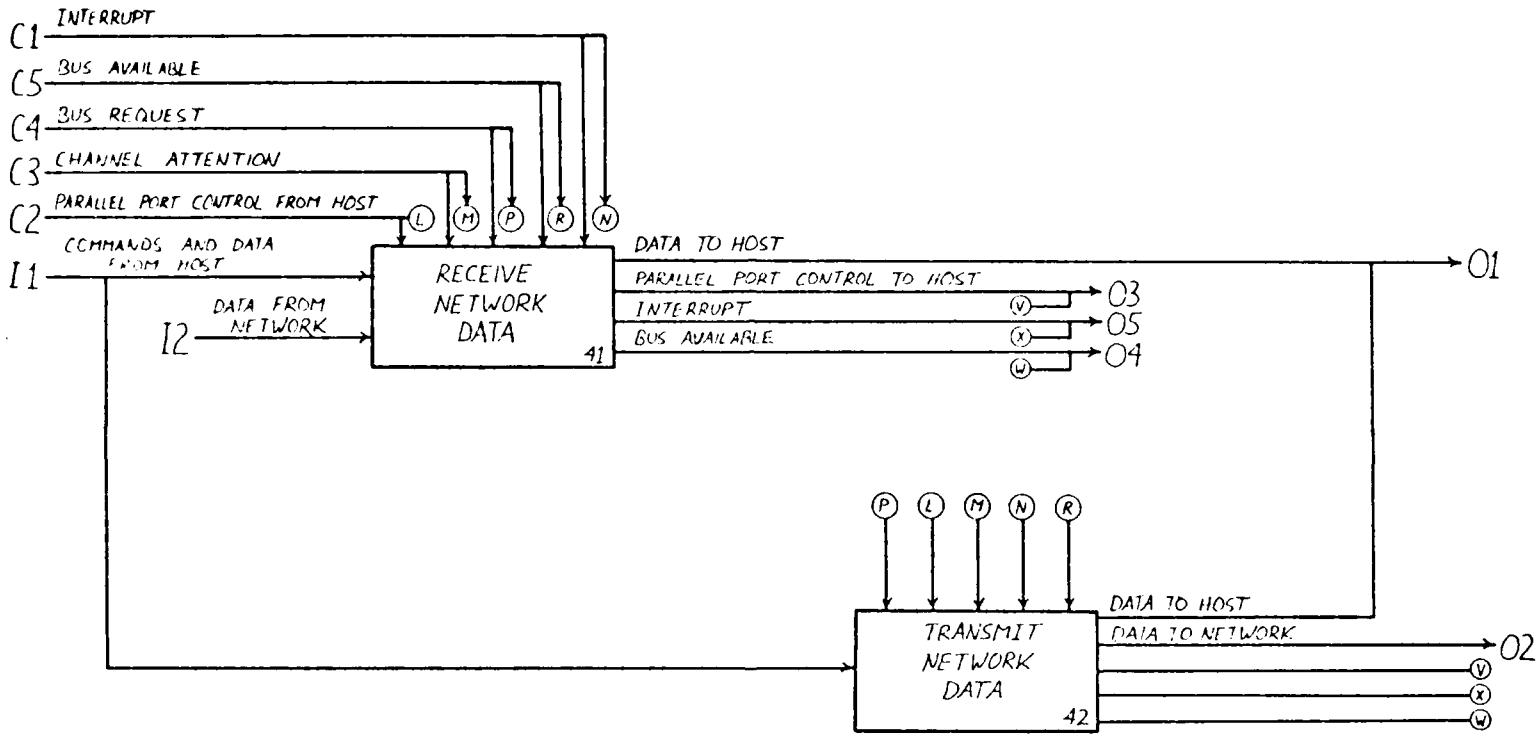
A412 Prepare SCB Start RU enables the NIB to receive data from the network. After requesting the available bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements while accepting commands and data. Note that the interrupt from a previous activity must be acknowledged before the RU is enabled. The RU is enabled by the host applying channel attention.

A413 (Receive Data) is an activity performed by the 82586 RU, and does not involve the host. It is included here only for completeness. When the 82586 detects an incoming data packet, it takes over the bus, receives the packet, and outputs an interrupt to notify the host of the reception.

A414 Transfer SCB Status To Host transfers the status of the reception to the host for verification of correctness. After requesting the available bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements. The result is the transfer of the status data to the host.

A415 Transfer Received Data To Host moves the received data from NIB memory to host memory. After requesting the available bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements, and the received data is transferred to the host.

Again, the order of execution is A411, A412, A413, A414, A415. Note, however, that A413 has priority.



NODE: A4 RECEIVE/TRANSMIT DATA

A4 Receive/Transmit Data

Abstract: This diagram decomposes the activity of Network Data Exchange into its major functions.

Network Data Exchange is composed of activities which perform the transmission and reception of data over the network. The two activities Receive Network Data and Transmit Network Data are the real-time activities occurring during a simulation. One or the other may be active, but not simultaneously. Furthermore, activity Receive Network Data is of higher priority, such that if Transmit Network Data is active when data needs to be received, Transmit Network Data will be interrupted and completed at a later time. The means by which this is accomplished is through the bus available line. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from both activities. The interrupt output from each activity is typically the result of the host setting the interrupt bit in a command to the 82586, or, in the case of activity A41, it could be the result of a data reception. However, the interrupt could also be from an error. In any case, the interrupt must be acknowledged, and is shown as a control input.

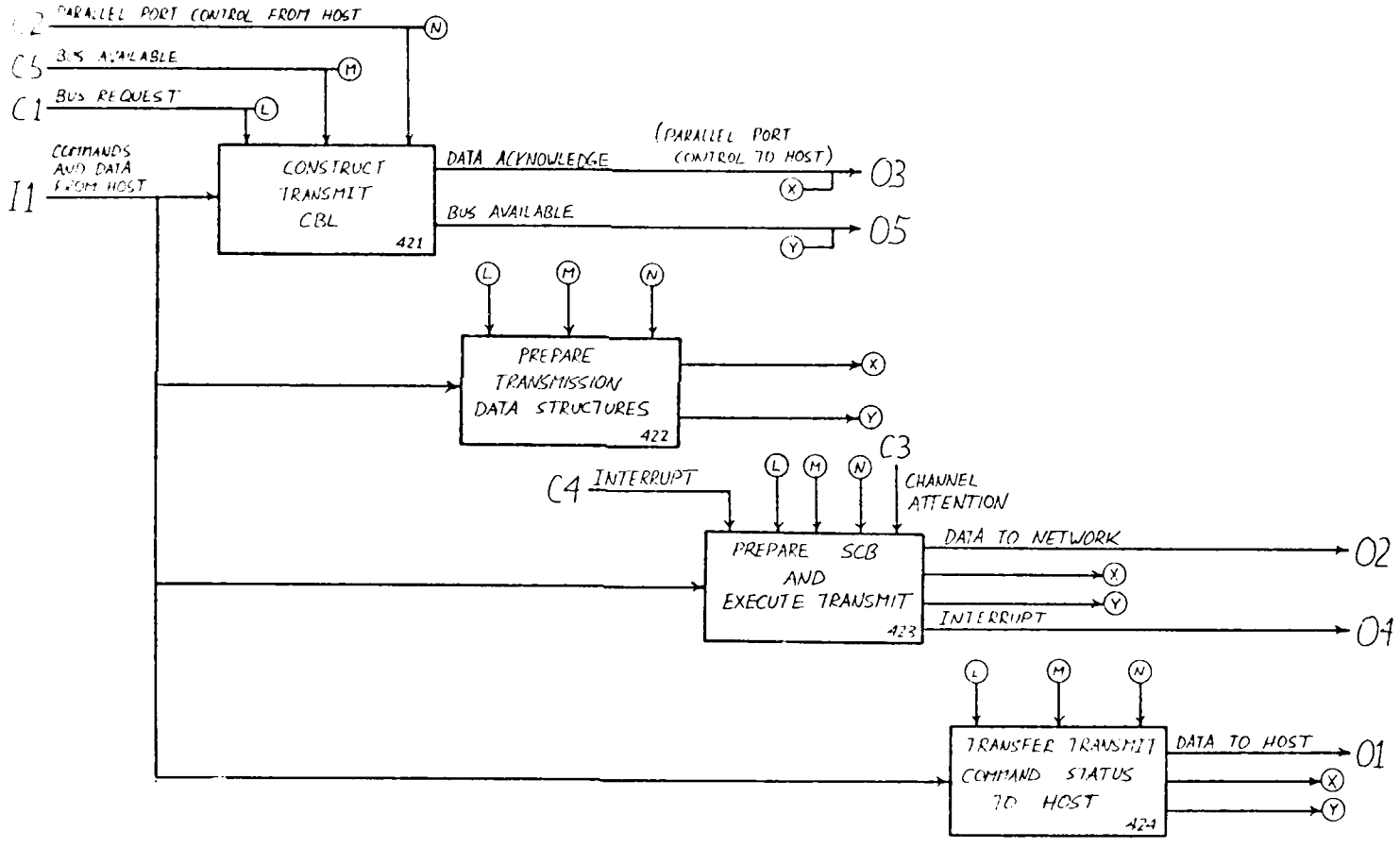
A41 Receive Network Data performs the task of data reception. After requesting the bus, the host uses parallel port controls to input commands and data from the host. The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with the command, and after data reception. The data is then transferred to the host.

A42 Transmit Network Data performs the task of data transmission. After requesting the bus, the host uses parallel port controls to input commands and data from the host. The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and data is transferred to the network. The NIB signals an interrupt when finished with execution. Finally, the Transmit command status word is transferred to the host to verify correct transmission.

Again, operation of the simulation switches between these two activities, with data reception having priority.

SAME AS NODE A312

NODE: A342 TRANSFER RESULTS TO HOST



NODE: A42 TRANSMIT NETWORK DATA

A421 Construct Transmit CBL

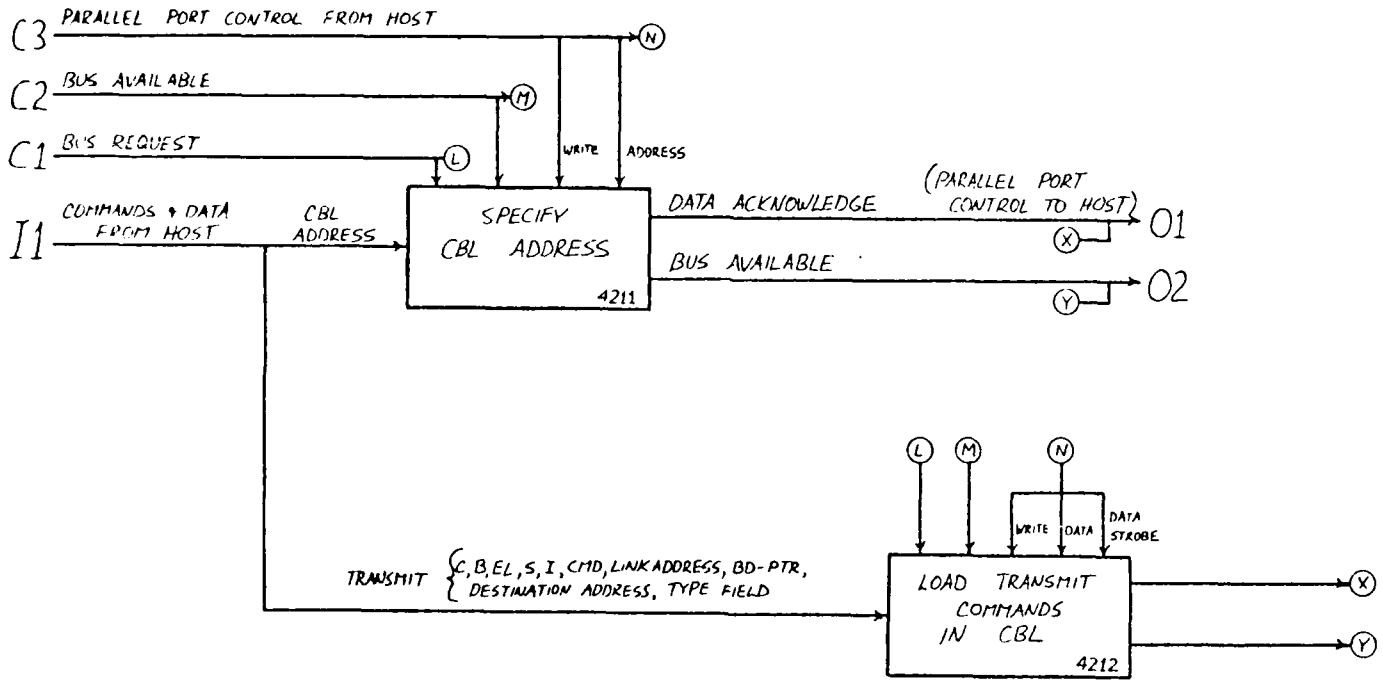
Abstract: This diagram decomposes the activity Construct Transmit CBL into its major functions.

Construct Transmit CBL handles the preparation of the Command Block List (CBL) containing one or more Transmit commands. Execution of activities A4211 and A4212 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the NIB bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from both activities.

A4211 Specify CBL Address performs the same function as activity A211. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4212 Load Transmit Commands In CBL handles the loading of one or more Transmit commands in a CBL which, when executed, will transmit data over the network. After requesting the available bus, the host loads the Transmit command parameters shown (according to the Transmit command data structure) into the CBL via the parallel port controls write, data, and data strobe. The 82586 responds with acknowledgements.

Again, the order of execution is A4211, A4212.



NODE: A421 CONSTRUCT TRANSMIT CBL

A422 Prepare Transmission Data Structures

Abstract: This diagram decomposes the activity Prepare Transmission Data Structures into its major functions.

Prepare Transmission Data Structures handles the preparation of the Transmit Buffer Descriptor (TBD) and the Data Buffer containing the data to be transmitted. Execution of A4221, A4222, A4223, and A4224 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the NIB bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from every activity.

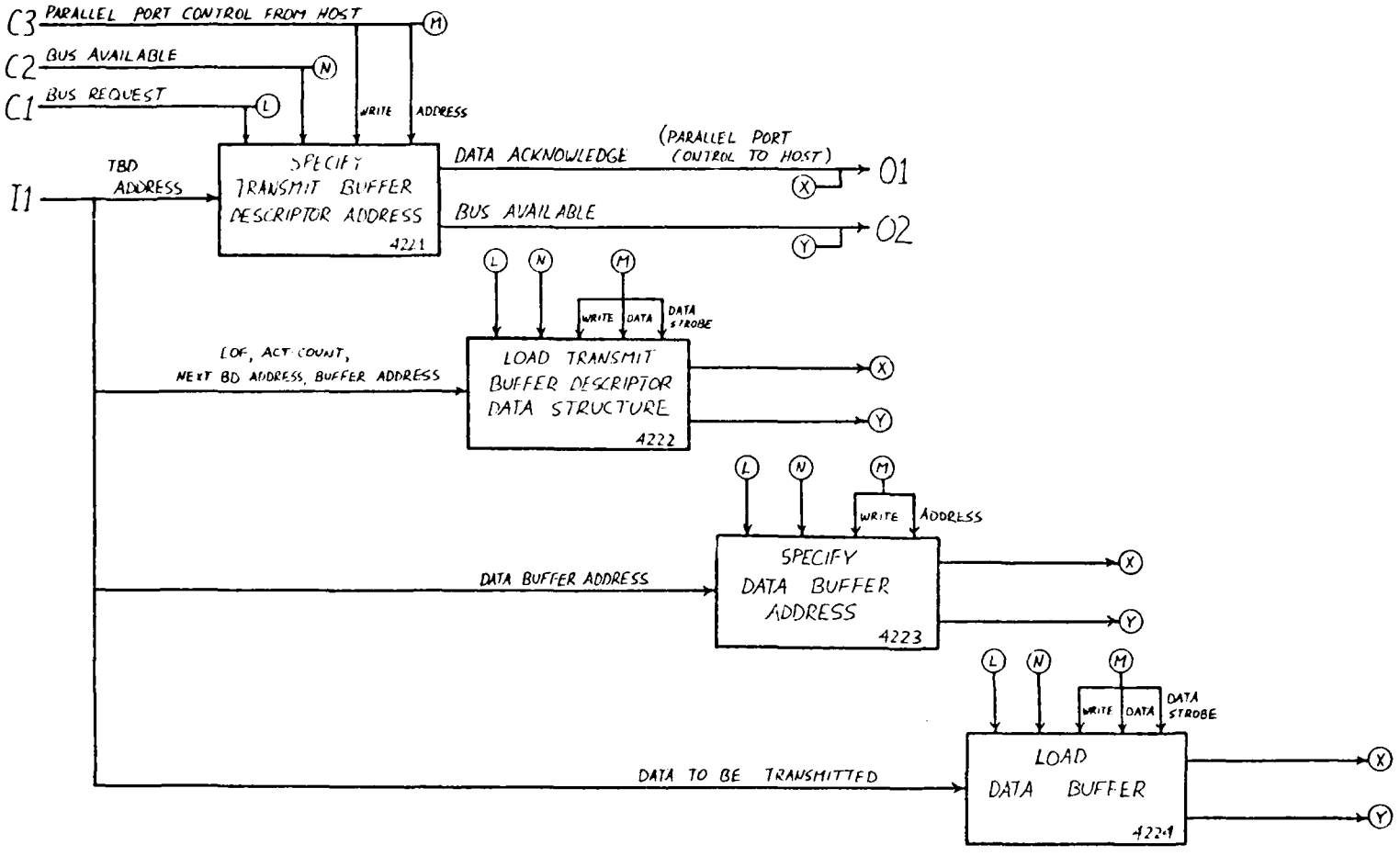
A4221 Specify Transmit Buffer Descriptor Address performs the same function as activity A32121. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4222 Load Transmit Buffer Descriptor Data Structure performs the same function as activity A32122. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4223 Specify Data Buffer Address performs the same function as activity A32123. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4224 Load Data Buffer performs the same function as activity A32124. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

Again, the order of execution is A4221, A4222, A4223, A4224.



NODE: A422 PREPARE TRANSMISSION DATA STRUCTURES

A423 Prepare SCB And Execute Transmit

Abstract: This diagram decomposes the activity Prepare SCB And Execute Transmit into its major functions.

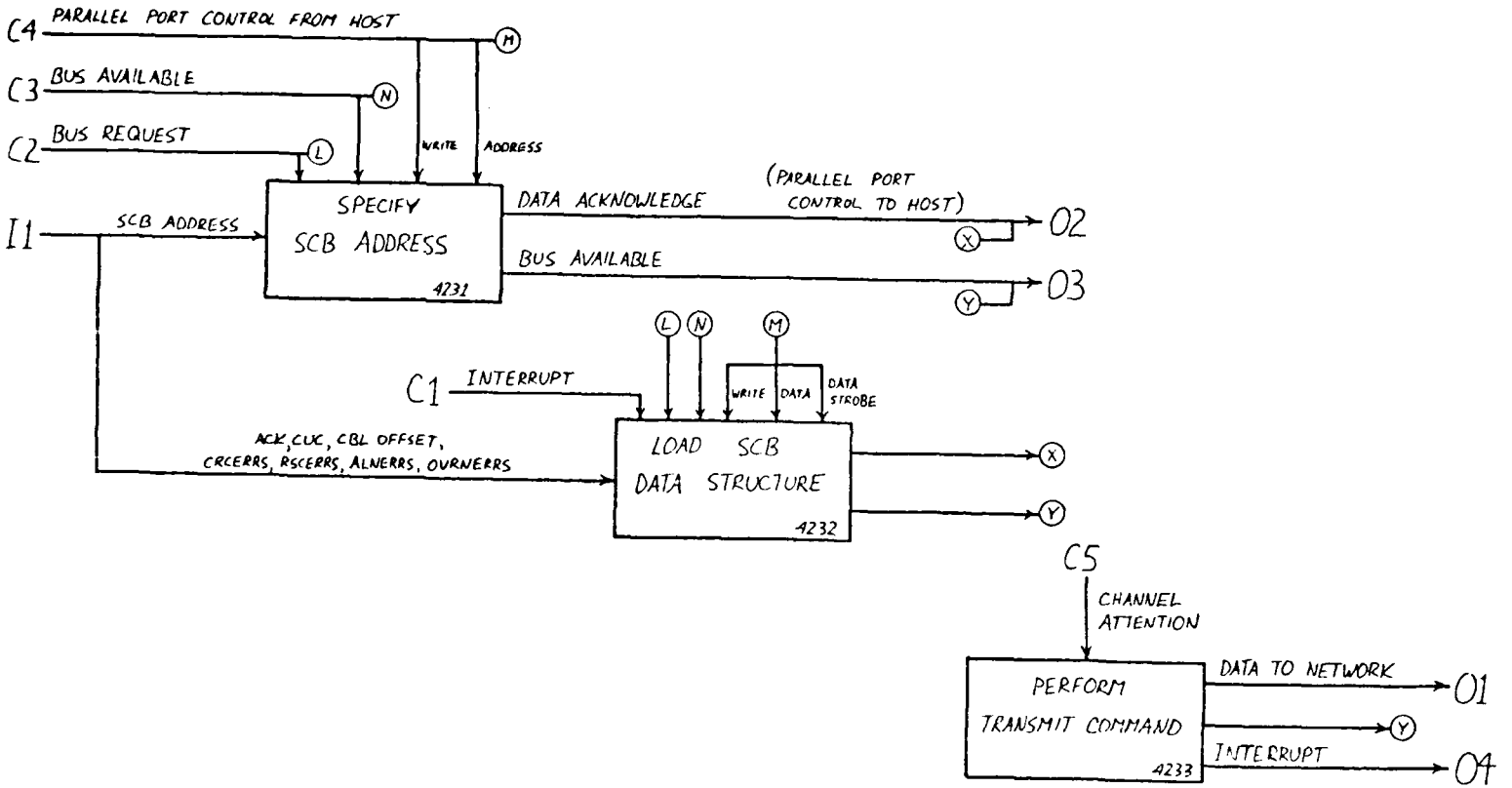
Prepare SCB And Execute Transmit handles the preparation of the System Control Block (SCB) in anticipation of executing the Transmit command CBL of activity A421. Execution of activities A4231, A4232, and A4233 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the NIB bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from every activity.

A4231 Specify SCB Address performs the same function as activity A213. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4232 Load SCB Data Structure performs the same function as activity A214. The only difference is that the activity may have to be repeated or continued later if the 82586 takes over the NIB bus during its execution.

A4233 Perform Transmit Command is the execution of the Transmit command CBL by the 82586. It is begun by the host applying channel attention to the NIB. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the Transmit command). The end result is the transmission of the data over the network.

Again, the order of execution is A4231, A4232, A4233.



NODE: A423 PREPARE SCB AND EXECUTE TRANSMIT

A424 Transfer Transmit Command Status To Host

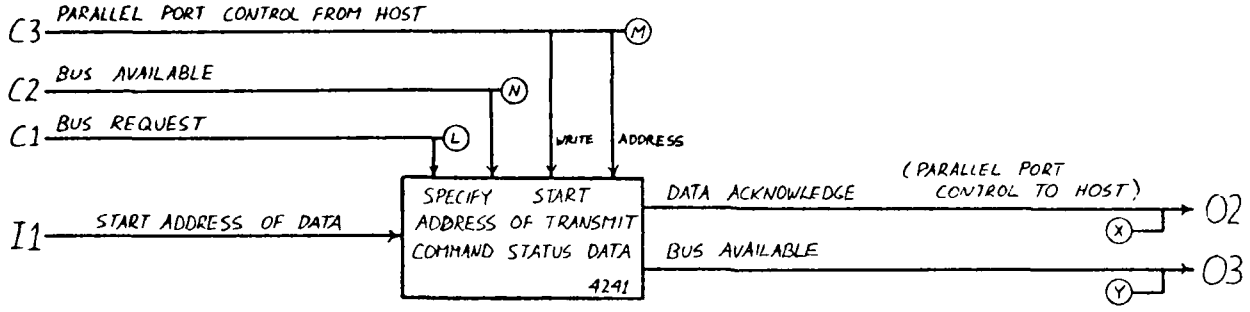
Abstract: This diagram decomposes the activity Transfer Transmit Command Status To Host into its major functions.

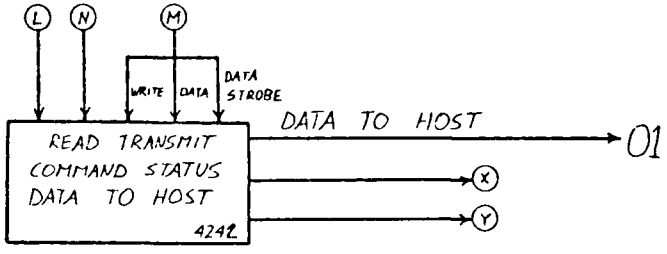
Transfer Transmit Command Status To Host handles the movement of the Transmit command status word from NIB memory to host memory. Execution of A4241 and A4242 is sequential as implied by their positioning from upper left to lower right. The host must monitor bus available while transferring commands and data to and from the NIB. Should the 82586 take over the bus, the host can either save its state and continue the interrupted activity later, or start the interrupted activity over. Therefore, bus available is shown as a control input and an output from activities A4241 and A4242.

A4241 Specify Start Address Of Transmit Command Status Data is performed by the host prior to transferring the Transmit Command status word to the host. After requesting the available bus, the host applies the start address of the Transmit command status word to the parallel port via the write and address controls. The NIB responds with data acknowledge.

A4242 Read Transmit Command Status Data To Host handles the function of transferring the status data. After requesting the bus, the host uses the parallel port controls read, data, and data strobe to step through NIB memory and read the status data. Data is transferred to the host, and the NIB responds with data acknowledgements.

Again, execution is in the order A4241, A4242.





NODE: A424 TRANSFER TRANSMIT COMMAND STATUS TO HOST

A5 Diagnose NIB Errors

Abstract: This diagram decomposes the activity Error Diagnostics into its major functions.

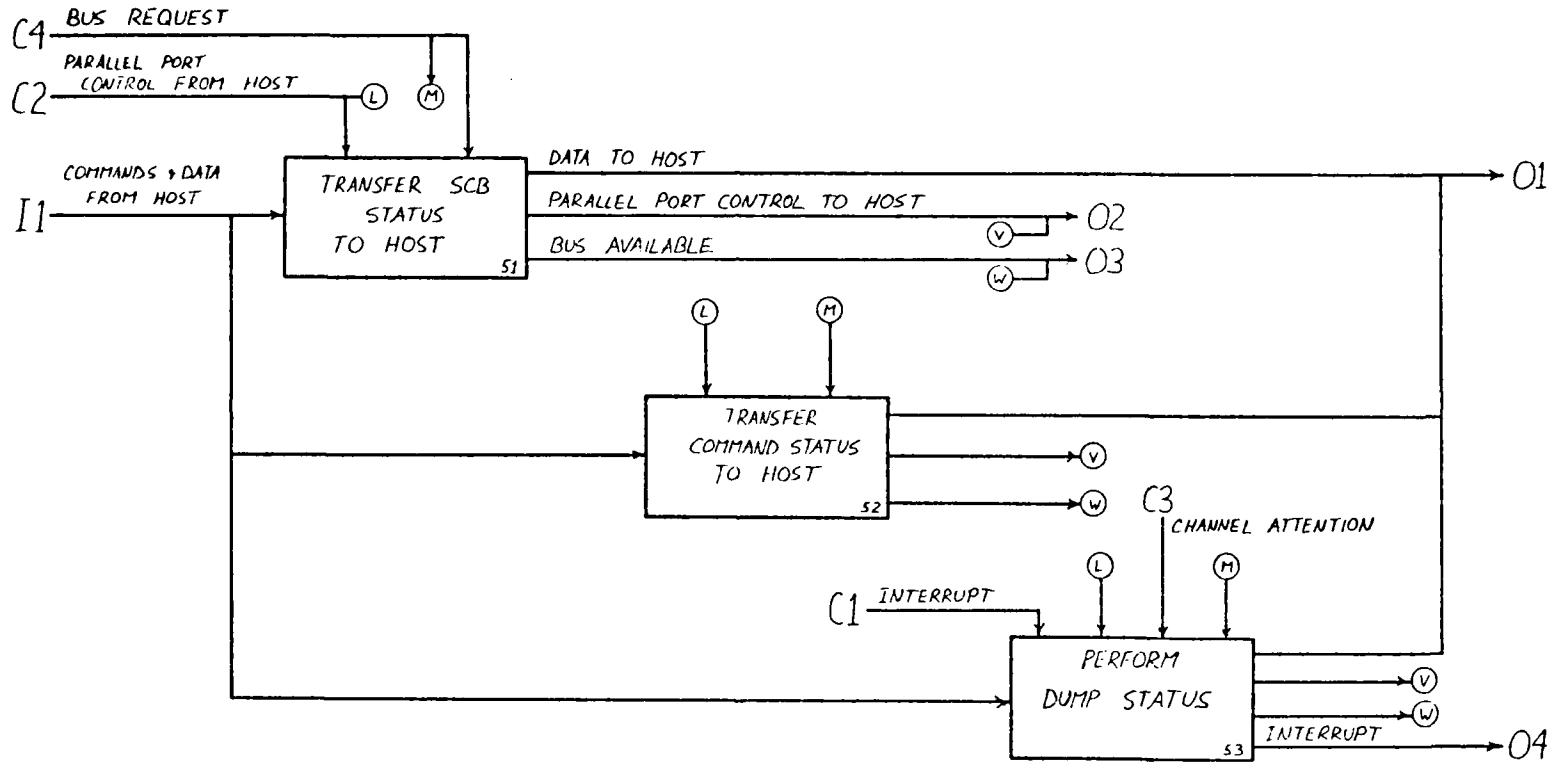
Error Diagnostics is composed of activities which transfer critical data from the NIB to the host to determine the cause of errors. Although the order of execution is implied by their positions in the diagram, activities A51, A52, and A53 can be executed in any order. However, the order shown is recommended as the information obtained goes from specific to general across activity A5. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Diagnose NIB Errors, the 82586 does take control of the bus during command execution (A53), but the host is not actively asking for the bus at this time, so there is no contention problem.

A51 Transfer SCB Status To Host transfers the System Control Block (SCB) status word to the host for analysis. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements. The result is the transfer of the status data to the host.

A52 Transfer Command Status To Host transfers the status word of the command experiencing the error to the host for analysis. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The NIB responds with data acknowledgements. The result is the transfer of the status to the host.

A53 Perform Dump Status executes the 82586 Dump Status command. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The host must also acknowledge the interrupt (shown as a control input) that was generated by the NIB upon completion of a previous activity. The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with command execution. The interrupt output is typically the result of the host setting the interrupt bit in the command to the 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by the host before the next activity prompted by a channel attention can be completed.

Again, the recommended order of execution is A51, A52, A53.



NODE: A5 DIAGNOSE NIB ERRORS

A51 Transfer SCB Status To Host

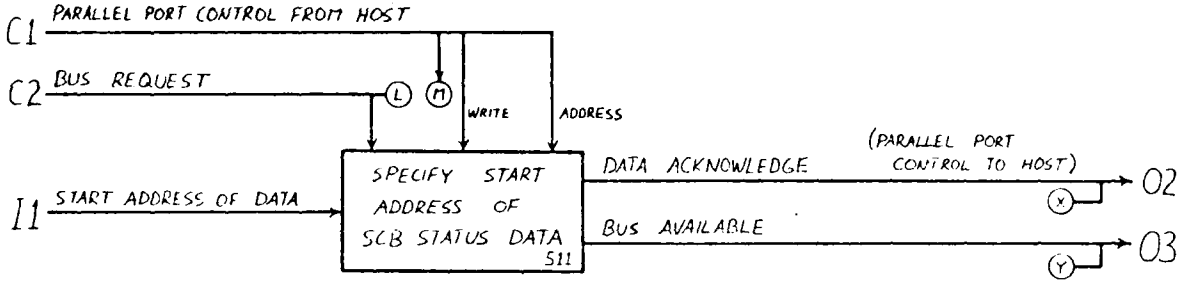
Abstract: This diagram decomposes the activity Transfer SCB Status To Host into its major functions.

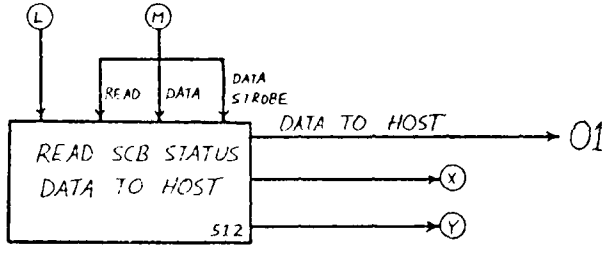
Transfer SCB Status To Host handles the movement of the System Control Block (SCB) status word from NIB memory to host memory. Execution of A511 and A512 is sequential as implied by their positioning from upper left to lower right. The bus available output from both activities is an indication as to whether or not the 82586 has taken control of the NIB bus. During activity Transfer SCB Status To Host, the 82586 does not ever take control, so there is no contention problem.

A511 Specify Start Address Of SCB Status Data is performed by the host prior to transferring the SCB status word to the host. After requesting the available bus, the host applies the start address of the SCB status word to the parallel port via the write and address controls. The NIB responds with data acknowledge.

A512 Read SCB Status Data To Host handles the function of transferring the status data. After requesting the bus, the host uses the parallel port controls read, data, and data strobe to step through NIB memory and read the status data. Data is transferred to the host, and the NIB responds with data acknowledgements.

Again, execution is in the order A511, A512.





NODE: A51 TRANSFER SCB STATUS TO HOST

A52 Transfer Command Status To Host

Abstract: This diagram decomposes the activity Transfer Command Status To Host into its major functions.

Transfer Command Status To Host handles the movement of the command status word of the command in error from NIB memory to host memory. Execution of A521 and A522 is sequential as implied by their positioning from upper left to lower right. The bus available output from both of these activities is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Transfer Command Status To Host, the 82586 does not ever take control, so there is no contention problem.

A521 Specify Start Address Of Command Status Data is performed by the host prior to transferring the command status word to the host. After requesting the bus, the host applies the start address of the command status word to the parallel port via the write and address controls. The NIB responds with data acknowledge.

A522 Read Command Status Data To Host handles the function of transferring the status data. After requesting the bus, the host uses the parallel port controls read, data, and data strobe to step through NIB memory and read the status data. Data is transferred to the host, and the NIB responds with data acknowledgements.

Again, execution is in the order A521, A522.

AD-A152 242 DESIGN AND SPECIFICATION OF A LOCAL AREA NETWORK ARCHITECTURE FOR USE IN.. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. L R MAKI

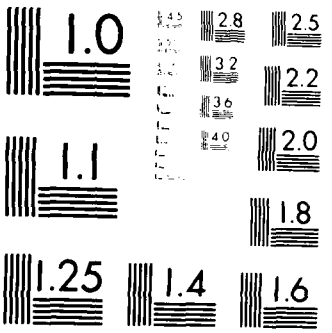
3/3

UNCLASSIFIED SEP 84 AFIT/GCS/ENG/84S-3 F/G 9/5 NL

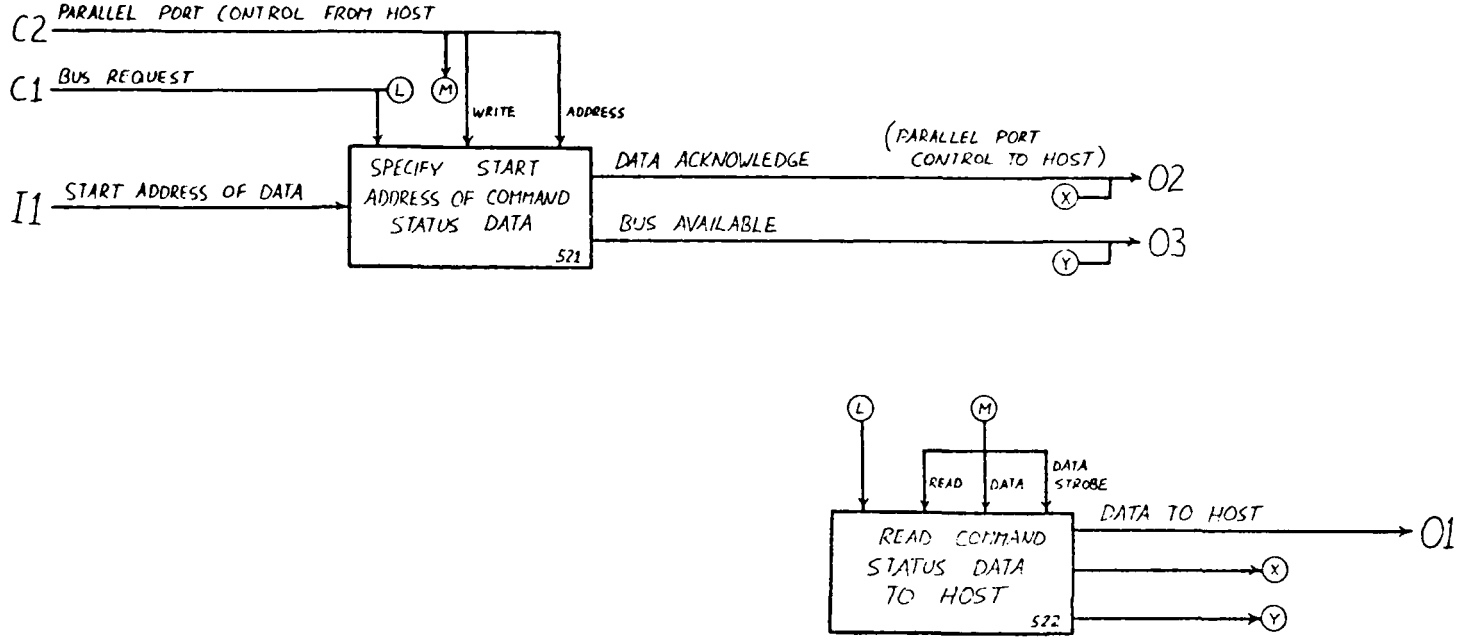
END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



NODE: A52 TRANSFER COMMAND STATUS TO HOST

A53 Perform Dump Status

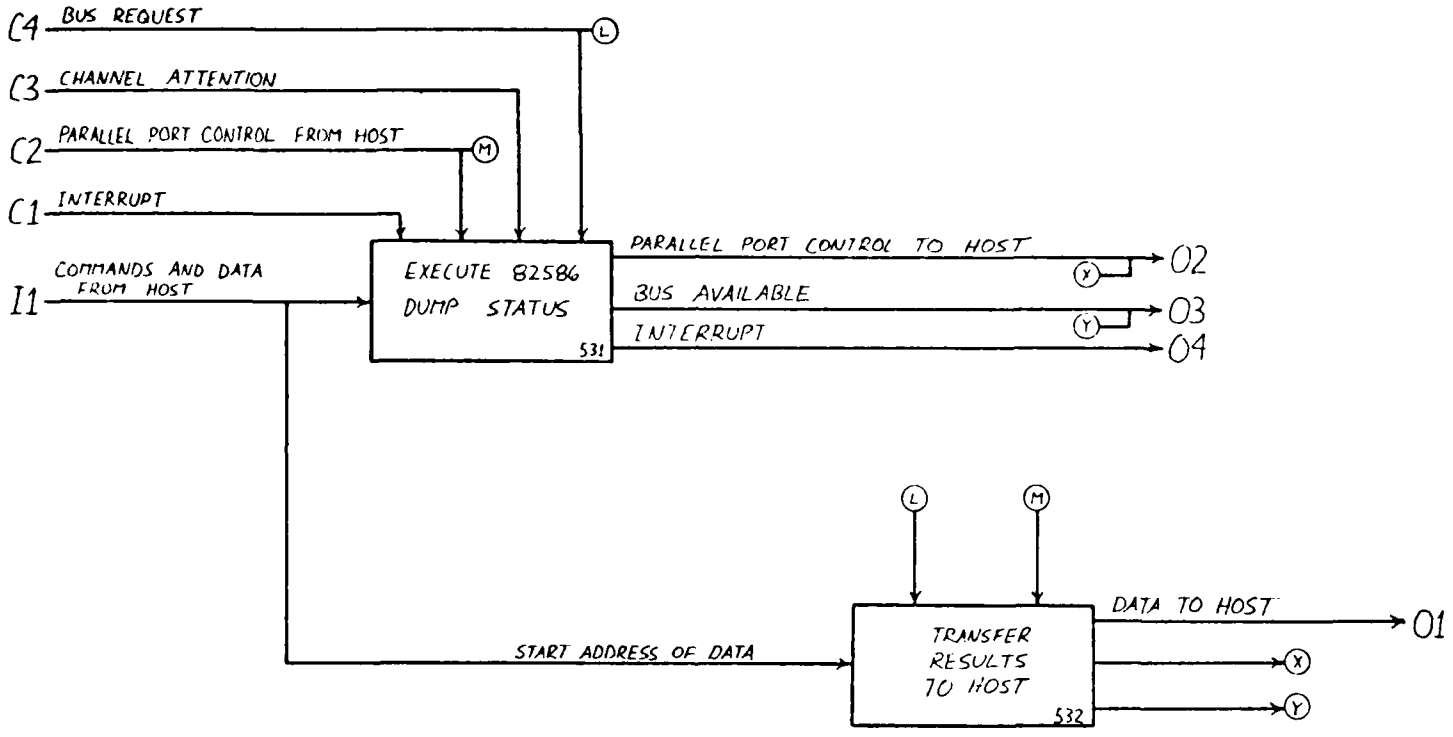
Abstract: This diagram decomposes the activity Perform Dump Status into its major functions.

Perform Dump Status is composed of activities which set up and execute the 82586 Dump Status command and transfer the results to the host. The order of execution of activities A531 and A532 is sequential as implied by their positioning from upper left to lower right.

A531 Execute 82586 Dump Status performs a dump into NIB memory of the contents of various 82586 registers. After requesting the bus, the host uses parallel port controls to input commands and data for executing this activity. The host must also acknowledge the interrupt (shown as a control input) that was generated by the NIB upon completion of a previous activity. The NIB responds with parallel port controls while accepting commands and data. The host starts command execution by applying channel attention, and the NIB signals an interrupt when finished with execution. The bus available output from this activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Execute 82586 Dump Status, the 82586 does take control of the bus during command execution. However, the host is not actively asking for the bus at this time, so there is no contention problem. The interrupt output from this activity is typically the result of the host setting the interrupt bit in the command to he 82586, although it could be the result of an error. For either reason, the interrupt must be acknowledged by te host before he next activity prompted by a channel attention can be completed.

A532 (Identical to activity A312)

Again, the order of execution is A531, A532.



NODE: A53 PERFORM DUMP STATUS

A531 Execute 82586 Dump Status

Abstract: This diagram decomposes the activity Execute 82586 Dump Status into its major functions.

Execute 82586 Dump Status requires the host to create in NIB memory a Command Block List (CBL) containing the Dump Status command. The 82586 System Control Block (SCB) is then prepared so that, when channel attention is applied, the 82586 knows to execute the Dump Status command. Execution of A5311, A5312, A5313, A5314, and A5315 is sequential as implied by their positioning from upper left to lower right. The bus available output from each activity is an indication as to whether or not the 82586 has taken control of the NIB bus. During the activity Execute 82586 Dump Status, the 82586 does take control as it executes the command. However, the host is not actively asking for the bus at this time, so there is no contention problem.

A5311 (Identical to activity A211)

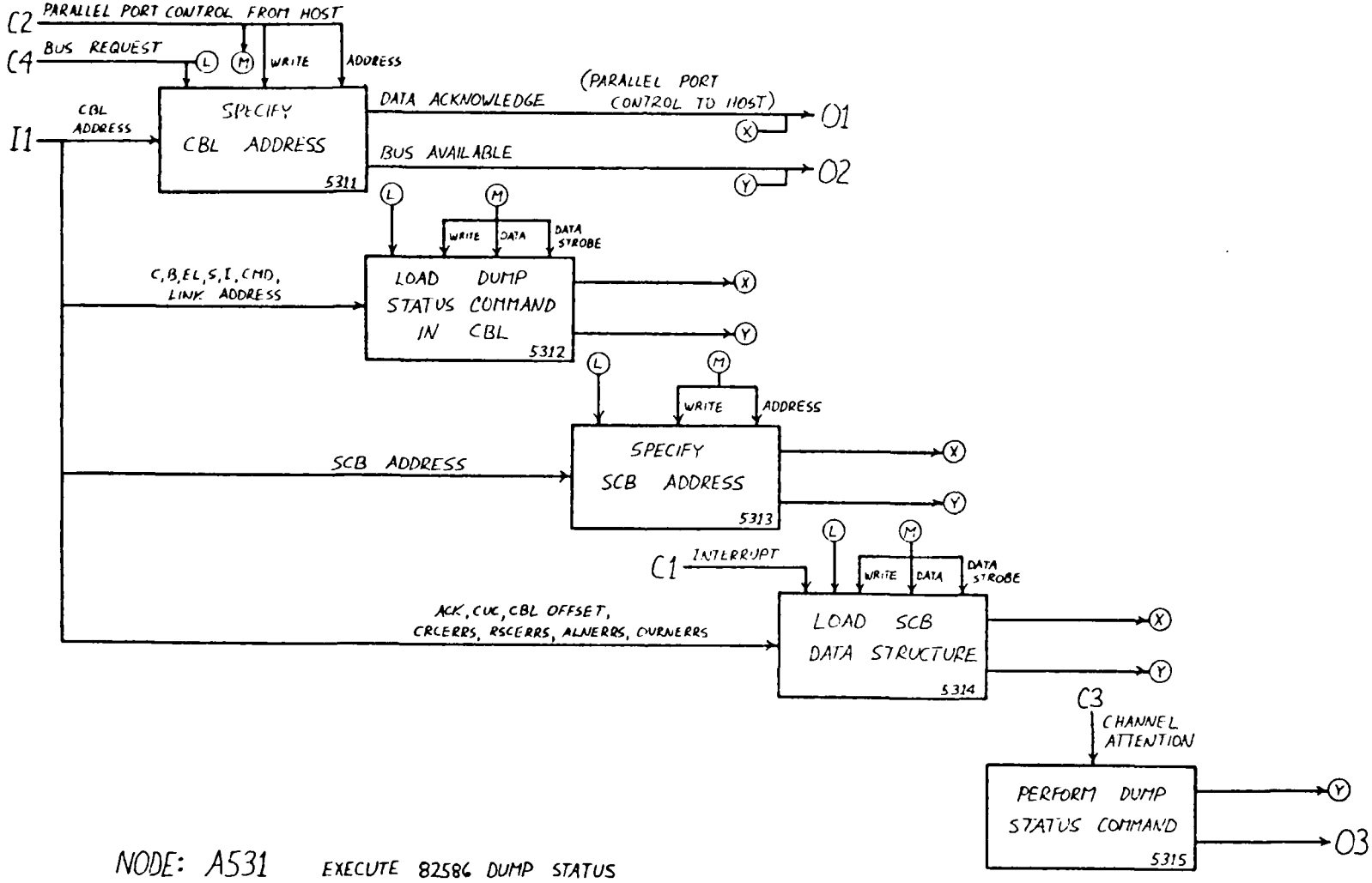
A5312 Load Dump Status Command In CBL loads the Dump Status command parameters shown as inputs into the CBL located by activity A5311. After requesting the bus, the host loads the Dump Status command parameters (according to the Dump Status command data structure) into the CBL via the parallel controls write, data, and data strobe. The 82586 responds with data acknowledgements.

A5313 (Identical to activity A213)

A5314 (Identical to activity A214)

A5315 Perform Dump Status Command is the execution of the Dump Status command by the 82586. It is begun by the host applying channel attention to the NIB. The NIB responds with an interrupt when command execution is completed (if the I bit is set in the Dump Status CBL).

Again, the order of execution is A5311, A5312, A5313, A5314, A5315.



NODE: A531 EXECUTE 82586 DUMP STATUS

A532 Transfer Results To Host

Abstract: This activity performs the same function as activity A312.

SAME AS NODE A312

NODE: A532 TRANSFER RESULTS TO HOST

TABLE D.2

Glossary/Data Dictionary

ACK - Acknowledgement bits set by the host in the System Control Block.

ACT-COUNT - Indicates the number of bytes that hold information for the current buffer.

Address - The parallel port control specifying that an address is on the 16 bidirectional lines.

ALNERRS - The number of misaligned frames discarded because of CRC errors, initialized to zero by the host, updated by the 82586.

B - Indicates when set that the 82586 is currently executing the command.

BD-PTR - The pointer to the Buffer Descriptor.

Buffer Address - The starting address of the memory area that contains the data to be sent.

Bus Available - Signal to the host indicating the availability of the NIB bus.

Bus Request - Signal to the NIB indicating the host wants control of the NIB bus.

Busy - Indicates that the 82586 is being initialized.

Byte Count - The number of bytes in the configure command to be changed.

C - Indicates the execution status of a command; set following execution.

CBL Address - The address of the Command Block List.

CBL Offset - The offset portion of the Command Block list address.

Channel Attention - Signal applied to the NIB by the host to execute commands in a Command Block list.

CMD - The command opcode.

Commands And Data From Host - Information supplied to the NIB from the host necessary for activity execution.

TABLE D.2

Glossary/Data Dictionary (Continued)

Configuration - The process of tailoring the 82586 for the application.

Configuration Parameters - The parameters in the Configure command allowing the 82586 to be tailored to the application.

Configure For Internal Loopback - The Configure command set for internal loopback.

Configure For External Loopback - The Configure command set for external loopback.

CRCERRS - The number of aligned frames discarded because of CRC errors, initialized to zero by the host, updated by the 82586.

CUC - The opcode command to the Command Unit in the System Control Block.

Data - The parallel port control specifying that data is on the 16 bidirectional lines.

Data Acknowledge - A parallel port control having a dual purpose. In the read mode, it asserts a pulse while the NIB holds the data valid. In the write mode, it's pulse acknowledges receipt of the data or address.

Data From Network - Information received from the network side of the NIB.

Data Strobe - The data synchronization strobe having a dual purpose. In the read mode, it signals the NIB to output the data at the next sequential address. In the write mode, it signals the NIB to strobe in the data on the bus into the present memory address location and then increment the address to the next sequential location.

Data To Be Transmitted - Data to be transmitted over the network.

Data To Host - Information sent to the host from the NIB.

Data To Network - Information sent to the network from the NIB.

Destination Address - The address to which data is to be sent.

EL - Indicates when set that the command is the last on the CBL.

TABLE D.2

Glossary/Data Dictionary (Continued)

EOF - Indicates when set that the TBD is the last associated with the frame being transmitted.

Error Diagnostics - The process of determining the cause of an error which occurred during System Diagnostics or network Data Exchange.

F - Indicates when set that the RBD has already been used.

I - Indicates when set that the 82586 will generate an interrupt after execution of the command is completed.

Individual Address - The unique address by which the NIB will be recognized.

Initialization - The process of initializing the 82586 and getting it to a standard state.

Interrupt - Signal to the host indicating one or more of the following: the command just executed has the interrupt bit set; a data packet has been received; the 82586 Receive Unit is not ready; the 82586 Command Unit is not ready.

ISCP Address - The address of the 82586 Intermediate System Control Pointer.

Link Address - A pointer to the next command on the CBL.

Multicast Addresses - The multicast addresses by which the NIB will be recognized.

Network Data Exchange - The real-time process of receiving and transmitting data over the network.

Next BD Address - The offset portion of the address of the next TBD on the TBD list.

OVRNERRS - The number of frames that are known to be lost due to a lack of local system bus availability, set to zero by the host, updated by the 82586.

Parallel Port Control From Host - Handshaking communication controls to enable data transfer between the host and the NIB.

RBD Address - The address of the Receive Buffer Descriptor.

TABLE D.2

Glossary/Data Dictionary (Concluded)

Read - The parallel port control specifying data is to be read from the NIB.

Reset - Signal applied by the host to terminate activities, just prior to initialization.

RFA Offset - The offset portion of the Receive Frame Area address.

RFD Address - The address of the Receive Frame Descriptor.

RSCERRS - The number of good frames discarded because there were no resources to receive them, initialized to zero by the host, updated by the 82586.

RUC - The opcode command to the 82586 Receive Unit in the System Control Block.

S - Indicates when set that the 82586 Command Unit should be suspended following execution of the current command.

SCB Address - The SCB base address.

SCB Base - The base portion of the System Control Block address.

SCB Offset - The offset portion of the System Control Block address.

SCP Address - The address of the 82586 System Configuration Pointer.

Start Address Of Data - The starting address in NIB memory of data to be examined by the host.

Sysbus - The word size the 82586 will use for transfers (8 or 16 bits).

System Diagnostics - The process of checking the NIB for correct operation.

TBD Address - The address of the Trasmit Buffer Descriptor.

Type Field - A user-defined data field, typically used to indicate the type of data contained in a packet.

Write - The parallel port control specifying data is to be written to the NIB.

Bibliography

Allan, 1982a. Allan, Roger. "Designer's Reference: A Guide to Standards for Buses, LANs, Ergonomics," Electronic Design, 30 (26): 107-112 (December 23, 1982)a.

Allan, 1982b. -----. "Local Networks Broad Standards, Many Implementations on the Way," Electronic Design, 30 (20): 87-101 (September 30, 1982)b.

Arthurs and Stuck, 1982. Arthurs, E. and B. W. Stuck. "A Theoretical Performance Analysis of Polling and Carrier Sense Collision Detection Communication Systems," Local Computer Networks, edited by P. C. Ravasio, G. Hopkins, and N. Naffah. North-Holland Publishing Company, 1982.

Belden Corporation, 1983. 882 REV-2. Belden Electronics Wire and Cable. Product Catalog. Belden Corporation, Electronic Wire and Cable Division, Richmond, IN, 1983.

Benken, 1984. Benken, Lt Richard P., Digital Design Engineer for the FCDL LAN Interface. Personal Interviews and Notes. AFWAL/FIGD, Wright-Patterson AFB OH, January 1983 through May 1984.

Blair and Shepherd, 1982. Blair, Gordon S. and Doug Shepherd. "A Performance Comparison of Ethernet and the Cambridge Digital Communication Ring," Computer Networks, 6 (2): 105-113 (1982).

Burskyin, 1982. Burskyin, Dave. "Special Report: Silicon Ousting Software in Network Systems," Electronic Design, 30 (20): 73-82 (September 30, 1982).

Bux, 1981. Bux, Werner. "Local Area Subnetworks: A Performance Comparison," IEEE Transactions on Communications, COM-29 (10): 1465-1473 (October 1981).

Digital Equipment Corporation and others, 1980. Digital Equipment Corporation, Intel Corporation, Xerox Corporation. The Ethernet: A Local Area Network: Data Link Layer and Physical Layer Specifications. Version 1.0, September 30, 1980.

Franta and Chlamtac, 1981. Franta, W. R. and Imrich Chlamtac. Local Networks. Lexington: D. C. Heath and Company, 1981.

Gould, 1980. 303-000270-200. High-Speed Data Interface, Model 9131. Technical Manual, Description of High-Speed Data Interface for SEL Computers. Gould Electronics & Electrical Products, February, 1980.

Intel Corporation, 1982. 210783-001. The Complete VLSI Solution. Description of 82586 and 82501 LAN Chip Set. Intel Corporation, Santa Clara, CA, 1982.

Intel Corporation, 1983. 210891-002. 82586 Reference Manual. Local Communications Controller Advance Information. Intel Corporation, Santa Clara, CA, January 1983.

Kirchoff and others, 1983. Kirchoff, Lt Arlen J., Program Manger, in cooperation with Luke R. Maki, Terry V. Christian, Karyl A. Adams, and Richard P. Benken. Simulation Requirements Specification Manned Combat Station. Report to the Division Chief of the Flight Control Division. Flight Control Development Laboratory, Air Force Wright Aeronautical Laboratories, 8 September 1983.

Lam 1980. Lam, Simon S. "A Carrier Sense Multiple Access Protocol for Local Networks," Computer Networks, 4 (1): 21-32 (February 1980).

Liu and others, 1982. Liu, Ming T., Wael Hilal, and Bernard H. Groomes. "Performance Evaluation of Channel Access Protocols for Local Computer Networks," Procedings of the 25th IEEE Computer Society International Conference. 417-426. IEEE Computer Society Press, Los Angeles, Fall 1982.

Metcalfe and Boggs, 1976. Metcalfe, Robert M. and David R. Boggs. "Ethernet: distributed Packet Switching for Local Computer Networks," Communications of the ACM, 19 (7): 395-404 (July 1976).

Reagan, 1983. Reagan, Philip H. Local Area Networks (Revised Edition). San Francisco: California Systems Group, 1983.

Shacham and Hunt, 1982. Schacham, Nachum and V. Bruce Hunt. "Performance Evaluation of the CSMA/CD (1-persistent) Channel-Access Protocol in Common-Channel Local Networks," Local Computer Networks, edited by P. C. Ravasio, G. Hopkins, and N. Naffah. North-Holland Publishing Company, 1982.

Shoch and Hupp, 1980. Shoch, John F., and Jon A. Hupp. "Measured Performance of an Ethernet Local Network," Communications of the ACM, 23 (12): 711-721 (December 1980).

Stallings, 1984a. Stallings, William. Local Networks An Introduction. New York: Macmillan Publishing Company, 1984a.

Stallings, 1984b. -----. "Local Network Performance," IEEE Communications Magazine, 22 (2): 27-36 (February 1984)b.

Stuck, 1983. Stuck, Bart W. "Calculating the Maximum Mean Data Rate in Local Area Networks," Computer, 16 (5): 72-76 (May 1983).

Tobagi, 1982. Tobagi, Fouad A. "Carrier Sense Multiple Access with Message-Based Priority Functions," IEEE Transactions on Communications, COM-30 (1): 185-200 (January 1982).

Tobagi and Hunt, 1980. Tobagi, F. A., and V. B. Hunt. "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," Computer Networks, 4 (5): 245-259 (October/November 1980).

Vo-Dai, 1982. Vo-dai, Thien. "Through-Delay Analysis of the Non Slotted and Non Persistent CSMA-CD Protocol," Local Computer Networks, edited by P. C. Ravasio, G. Hopkins, and N. Naffah. North-Holland Publishing Company, 1982.

VITA

Mr. Luke R. Maki was born on 4 December 1955 in Akron, Ohio. He graduated from high school in Cuyahoga Falls, Ohio, in 1973 and attended the University of Akron from which he received the degree of Bachelor of Science in Mechanical Engineering in June 1978. Upon graduation, he ws employed by the Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, as a mechanical engineer to study the control of flight simulator motion systems. Other duties included the operation of an oculometer for the study of pilot eye scanning behavior, and coordinating simulator hardware modifications required for research simulation projects. He maintained this position at the Flight Dynamics Laboratory until entering the School of Engineering, Air Force Institute of Technology, in October 1982.

Permanent address: 3021 35th Avenue West
Seattle, Washington 98199

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GCS/ENG/84S-3	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Flight Control Division; 8b. OFFICE SYMBOL (If applicable): AFWAL/FIGD	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code): Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio 45433	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
62201F	2403	01	38

11. TITLE (Include Security Classification): See Box 19	
12. PERSONAL AUTHOR(S): Luke R. Maki, B.S., GS-12	
13a. TYPE OF REPORT: MS Thesis; 13b. TIME COVERED: FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day): 1984 September; 15. PAGE COUNT: 207
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	02	
09	05	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Local Area Networks, Ethernet, CSMA, CSMA/CD, Real-Time Flight Simulation, Network Performance

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: DESIGN AND SPECIFICATION OF A LOCAL AREA NETWORK ARCHITECTURE FOR USE IN REAL-TIME FLIGHT SIMULATION

Thesis Chairman: Walter D. Seward, Major, USAF
Associate Professor of Electrical Engineering

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Walter D. Seward, Major, USAF	513-255-3450	AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

This investigation examined the use of a Local Area Network (LAN) in the real-time environment of the Air Force Flight Dynamics Laboratory's Flight Control Development Laboratory (FCDL) flight simulation facility.

Using the requirements of the FCDL's Manned Combat Station (MCS) project as a guideline, a LAN based on the Ethernet protocol specification was identified as suitable for use in the real-time flight simulation facility. Both the architecture and the performance of the Ethernet protocol were examined to make this conclusion. The selection of the Intel 82586 Local Communications Controller chip (which defaults to the Ethernet specification), the in-house design and manufacture of a Network Interface Board (NIB) using this chip, and the standard operating procedure for FCDL simulations allowed a software design to be completed for the control of the NIB. A recommended course of action for the FCDL is made for the eventual implementation of the proposed LAN and software design.

END

FILMED

5-85

DTIC